

**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

Мельникова О.А.

Преподаватель

Кирияничков В.А.

Санкт-Петербург

2022

## Формулировка

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, потенциальный и граничный 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.

## 1. Расчет метрик вручную

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

В таблицах 1-3 представлены результаты подсчета числа типов операторов и операндов в программах на языке Паскаль, С и Assembler.

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

№	Оператор	Σ	№	Операнд	Σ	№	Операнд	Σ
1	()	29	1	0	1	25	pi2	3
2	*	17	2	0.0	5	26	small	3
3	+	9	3	0.01	1	27	sum	9
4	-	16	4	0.5	4	28	t	7
5	/	13	5	0.5772156	1	29	term	12
6	<	4	6	0.6366197	1	30	true	1
7	<>	1	7	1	14	31	ts	8
8	=	42	8	1.0	1	32	x	11
9	>=	1	9	1.0E-8	1	33	x2	4
10	abs	2	10	12	1	34	xx	6
11	bessy	2	11	2	3	35	y0	8
12	const	1	12	2.0	1	36	y1	8
13	for	1	13	3.1415926	1	37	ya	4
14	if	5	14	4	1	38	yb	5
15	ln	1	15	ans	5	39	yc	4
16	program	1	16	bessy	2	40	'Order? '	1
17	readln	2	17	besy	1	41	'Arg? '	1
18	real	1	18	done	4	42	'Y Bessel is '	1
19	repeat	4	19	euler	2			
20	sin	1	20	false	1			
21	sqrt	1	21	j	18			
22	trunc	1	22	n	5			
23	write	2	23	ordr	4			
24	writeln	2	24	pi	4			

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

№	Оператор	Σ	№	Операнд	Σ	№	Операнд	Σ
1	!	4	1	0	3	30	t	7
2	!=	1	2	0.0	5	31	t2	1
3	()	34	3	0.01	1	32	term	12
4	*	17	4	0.5	4	33	ts	8
5	+	9	5	0.5772156	1	34	x	11
6	+=	1	6	0.6366197	1	35	x2	4
7	,	23	7	1	16	36	xx	6
8	-	14	8	1.0	1	37	y0	8
9	/	13	9	1.0E-8	1	38	y1	8
10	<	4	10	12	1	39	ya	4
11	<=	1	11	2	3	40	yb	5
12	=	39	12	2.0	1	41	yc	4
13	==	2	13	3.1415926	1	42	"%f"	2
14	>=	1	14	4	1	43	"Arg? \n"	1
15	_&	2	15	a	1	44	"Order? \n"	1
16	_-	2	16	ans	5	45	"Y Bessel is %f \n"	1
17	abs	2	17	b	1	46	"\n"	1
18	bessy	2	18	cosa	1			
19	dowhile	4	19	done	3			
20	for	1	20	euler	2			
21	if	5	21	j	20			
22	log	1	22	n	5			
23	main	1	23	ordr	4			
24	printf	4	24	pi	4			
25	return	3	25	pi2	3			
26	scanf	2	26	sina	1			
27	sin	1	27	small	3			
28	sqrt	1	28	sum	9			
29	trunc	1	29	sum2	1			

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

№	Оператор	Σ	№	Оператор	Σ	№	Операнд	Σ	№	Операнд	Σ	№	Операнд	Σ
1	pushq	6	31	jne	5	1	0(%rbp)	2	31	%al	9	61	.L18	1
2	movq	43	32	movapd	2	2	48(%rsp)	2	32	\$56	4	62	.L22	1
3	subq	4	33	jnb	1	3	32(%rbp)	11	33	0	5	63	.L24	2
4	movsd	4	34	cvtsi2sdl	3	4	48(%rbp)	2	34	4	9	64	.L26	1
5	leaq	13	35	comisd	1	5	56(%rbp)	2	35	8	2	65	.L27	1
6	cvtss2sd	11	36	ucomiss	4	6	-16(%rbp)	11	36	16	1	66	.L29	1
7	divsd	3	37	jp	2	7	-4(%rbp)	15	37	48	3	67	.L8	1
8	pxor	27	38	comiss	5	8	40(%rbp)	11	38	56	2	68	.L9	1
9	movss	86	39	seta	2	9	-8(%rbp)	14	39	112	2	69	.LC0(%rip)	1
10	movl	23	40	xorl	3	10	112(%rsp)	1	40	128	1	70	.LC1(%rip)	1
11	addsd	3	41	testb	3	11	-44(%rbp)	3	41	%xmm0	179	71	.LC10(%rip)	3
12	mulss	12	42	cmovns	2	12	-52(%rbp)	4	42	%xmm1	68	72	.LC11(%rip)	1
13	addq	4	43	cvtss2sil	2	13	-64(%rbp)	3	43	%xmm2	17	73	.LC12(%rip)	1
14	popq	6	44	negl	2	14	-68(%rbp)	6	44	%xmm3	11	74	.LC13(%rip)	1
15	ret	4	45	mulsd	3	15	-12(%rbp)	10	45	%xmm4	10	75	.LC14(%rip)	1
16	cvtss2ss	4	46	subl	3	16	-36(%rbp)	13	46	%xmm5	8	76	.LC15(%rip)	2
17	addl	3	47	subsd	2	17	-1(%rax)	2	47	%rsp	1	77	.LC16(%rip)	1
18	cmpl	2	48	subsd	1	18	-72(%rbp)	7	48	3	14	78	.LC17(%rip)	1
19	je	2	49	movd	1	19	-28(%rbp)	3	49	-6	4	79	.LC2(%rip)	1
20	leal	2	50	Call __mingw_vfprintf	1	20	%rbp	20	50	"\12\0"	1	80	.LC3(%rip)	1
21	jmp	6	51	call log	1	21	%rbx	10	51	"Order? \12\0"	1	81	.LC4(%rip)	1
22	cvtsi2ss	8	52	call trunc	1	22	%rsp	9	52	"%f\0"	1	82	.LC5(%rip)	1
23	idivl	2	53	call sqrt	1	23	%rcx	10	53	"Arg? \12\0"	1	83	.LC7(%rip)	2
24	addss	4	54	call sin	1	24	%rdx	9	54	"Y Bessel is %f \12\0"	1	84	.LC8(%rip)	5
25	subss	7	55	call __main	1	25	%r8	4	55	.L10	1	85	.LC9(%rip)	3
26	xorps	2	56	call __mingw_vfscanf	1	26	%r9	2	56	.L11	2	86		
27	movaps	9	57	call printf	4	27	%rax	37	57	.L13	2	87		
28	divss	8	58	call scanf	2	28	%ecx	6	58	.L14	2	88		
29	cvtss2sil	2	59	call bessy	1	29	%eax	38	59	.L16	1	89		
30	comiss	5				30	%edx	6	60	.L17	1	90		

В таблице 4 представлены сводные результаты расчетных характеристик .

Таблица 4 – Результаты расчетных характеристик вручную

	<b>Паскаль</b>	<b>Си</b>	<b>Ассемблер</b>
Число уникальных операторов (n1):	<b>24</b>	29	59
Число уникальных операндов (n2):	<b>42</b>	46	85
Общее число операторов (N1):	159	195	376
Общее число операндов (N2):	178	187	673
Алфавит (n):	66	75	144
Экспериментальная длина программы (Nэ):	337	382	1049
Теоретическая длина программы (Nт):	336.516	394.965	891.874
Объем программы (V):	2036.96	2379.41	7521.25
Потенциальный объем (V*):	11,6096	11,6096	11.6096
Уровень программы (L):	0.005699	0.004879	0.00154358
Интеллект программы (I):	40.0526	40.3661	32.20119
Работа по программированию (E):	357393	487662,47	4872607.5
Время кодирования (T):	35739	48766,24	487260.75
Уровень языка программирования (Lam):	0.066169	0,56645	0.01792
Уровень ошибок (B):	3	3	8

## 2. Расчет метрик с помощью программы автоматизации

Для программы на Pascal:

Statistics for module output\_pas.lxm

```
=====
The number of different operators   : 24
The number of different operands    42
The total number of operators       159
The total number of operands        178
```

```
Dictionary      ( D ) : 66
Length          ( N ) : 337
Length estimation ( ^N ) : 336.516
Volume          ( V ) : 2036.96
Potential volume ( *V ) : 19.6515
Limit volume    (**V) : 38.2071
Programming level ( L ) : 0.00964745
Programming level estimation ( ^L ) : 0.0196629
Intellect       ( I ) : 40.0526
Time of programming ( T ) 11730
Time estimation   ( ^T ) : 5746.96
Programming language level (lambda) : 0.189587
Work on programming ( E ) 211140
Error            ( B ) : 1.18193
Error estimation  ( ^B ) : 0.678987
```

Table:

```
=====
Operators:
```

1	29	()
2	17	*
3	9	+
4	16	-
5	13	/
6	4	<
7	1	<>
8	42	=
9	1	>=
10	2	abs
11	2	bessy



12	1	const
13	1	for
14	5	if
15	1	ln
16	1	program
17	2	readln
18	1	real
19	4	repeat
20	1	sin
21	1	sqrt
22	1	trunc
23	2	write
24	2	writeln

Operands:

1	1	'Arg? '
2	1	'Order? '
3	1	'Y Bessel is '
4	1	0
5	5	0.0
6	1	0.01
7	4	0.5
8	1	0.57721566
9	1	0.63661977
10	14	1
11	1	1.0
12	1	1.0E-8
13	1	12
14	3	2
15	1	2.0
16	1	3.1415926
17	1	4
18	5	ans
19	2	bessy
20	1	besy
21	4	done
22	2	euler
23	1	false
24	18	j
25	5	n
26	4	ordr
27	4	pi
28	3	pi2
29	3	small
30	9	sum

31	7	t
32	12	term
33	1	true
34	8	ts
35	11	x
36	4	x2
37	6	xx
38	8	y0
39	8	y1
40	4	ya
41	5	yb
42	4	yc

#### Summary:

```
=====
The number of different operators : 24
The number of different operands 42
The total number of operators    159
The total number of operands     178
```

```
Dictionary      ( D ) : 66
Length          ( N ) : 337
Length estimation ( ^N ) : 336.516
Volume          ( V ) : 2036.96
Potential volume ( *V ) : 19.6515
Limit volume    (**V) : 38.2071
Programming level ( L ) : 0.00964745
Programming level estimation ( ^L ) : 0.0196629
Intellect      ( I ) : 40.0526
Time of programming ( T ) : 11730
Time estimation ( ^T ) : 5746.96
Programming language level (lambda) : 0.189587
Work on programming ( E ) : 211140
Error              ( B ) : 1.18193
Error estimation   ( ^B ) : 0.678987
```

#### **Для программы на языке C:**

#### Statistics for module output.lxm

```
=====
The number of different operators 29
The number of different operands 46
The total number of operators    195
The total number of operands     187
```

Dictionary ( D) : 75  
 Length ( N) : 382  
 Length estimation ( ^N) : 394.965  
 Volume ( V) : 2379.41  
 Potential volume (\*V) : 19.6515  
 Limit volume (\*\*V) : 38.2071  
 Programming level ( L) : 0.00825898  
 Programming level estimation ( ^L) : 0.0169648  
 Intellect ( I) : 40.3661  
 Time of programming ( T) : 16005.5  
 Time estimation ( ^T) : 8056.45  
 Programming language level (lambda) : 0.162301  
 Work on programming ( E) : 288100  
 Error ( B) : 1.45403  
 Error estimation ( ^B) : 0.793136

Table:

=====

Operators:

1	4	!
2	1	!=
3	34	()
4	17	*
5	9	+
6	1	+=
7	23	,
8	14	-
9	13	/
10	4	<
11	1	<=
12	39	=
13	2	==
14	1	>=
15	2	_&
16	2	_-
17	2	abs
18	2	bessy
19	4	dowhile
20	1	for
21	5	if
22	1	log
23	1	main
24	4	printf
25	3	return
26	2	scanf
27	1	sin

28	1	sqrt
29	1	trunc

Operands:

1	2	"%f"
2	1	"Arg? \n"
3	1	"Order? \n"
4	1	"Y Bessel is %f \n"
5	1	"\n"
6	3	0
7	5	0.0
8	1	0.01
9	4	0.5
10	1	0.57721566
11	1	0.63661977
12	16	1
13	1	1.0
14	1	1.0E-8
15	1	12
16	3	2
17	1	2.0
18	1	3.1415926
19	1	4
20	1	a
21	5	ans
22	1	b
23	1	cosa
24	3	done
25	2	euler
26	20	j
27	5	n
28	4	ordr
29	4	pi
30	3	pi2
31	1	sina
32	3	small
33	9	sum
34	1	sum2
35	7	t
36	1	t2
37	12	term
38	8	ts
39	11	x
40	4	x2
41	6	xx
42	8	y0
43	8	y1
44	4	ya
45	5	yb

Summary:

```
=====
The number of different operators      : 29
The number of different operands      46
The total number of operators          195
The total number of operands          187
Dictionary        ( D)    75
Length            ( N)   382
Length estimation  ( ^N)   : 394.965
Volume            ( V)    : 2379.41
Potential volume   ( *V)   : 19.6515
Limit volume       (**V)   : 38.2071
Programming level  ( L)    : 0.00825898
Programming level estimation ( ^L)   : 0.0169648
Intellect          ( I)    : 40.3661
Time of programming ( T)    : 16005.5
Time estimation     ( ^T)   : 8056.45
Programming language level (lambda) : 0.162301
Work on programming ( E)    288100
Error              ( B)    : 1.45403
Error estimation    ( ^B)   : 0.793136
```

## **Вывод**

Метрические характеристики программ, написанных на языках Си и Паскаль, выглядят похожим образом так как имеют схожую структуру. Так как Ассемблер является языком низкого уровня, то характеристики программы, написанной на языке Ассемблер, значительно отличаются. Характеристики были посчитаны вручную и автоматически.

## ПРИЛОЖЕНИЕ А.

### КОД ПРОГРАММЫ НА ЯЗЫКЕ ПАСКАЛЬ

```
program besy;

var    x,ordr  : real;
       done    : boolean;

function bessy(x,n: real): real;
const  small   = 1.0E-8;
       euler   = 0.57721566;
       pi      = 3.1415926;
       pi2     = 0.63661977;
var    j       : integer;

       x2,sum,t,
       ts,term,xx,y0,y1,
       ya,yb,yc,ans      : real;

begin
  if x<12 then
    begin
      xx:=0.5*x;
      x2:=xx*xx;
      t:=ln(xx)+euler;
      sum:=0.0;
      term:=t;
      y0:=t;
      j:=0;
      repeat
        j:=j+1;
        if j<>1 then sum:=sum+1/(j-1);
        ts:=t-sum;
        term:=-x2*term/(j*j)*(1-1/(j*ts));
        y0:=y0+term
      until abs(term)<small;
      term:=xx*(t-0.5);
      sum:=0.0;
      y1:=term;
      j:=1;
      repeat
        j:=j+1;
        sum:=sum+1/(j-1);
        ts:=t-sum;
        term:=(-x2*term)/(j*(j-1))*((ts-0.5/j)/(ts+0.5/(j-1)));
        y1:=y1+term
      until abs(term)<small;
      y0:=pi2*y0;
      y1:=pi2*(y1-1/x);
      if n=0.0 then ans:=y0
      else if n=1.0 then ans:=y1
      else
        begin
          ts:=2.0/x;
          ya:=y0;
          yb:=y1;
          for j:=2 to trunc(n+0.01) do
            begin
              yc:=ts*(j-1)*yb-ya;
              ya:=yb;
              yb:=yc
```

```

        end;
    ans:=yc
    end;
    bessy:=ans;
end
else
    bessy:=sqrt(2/(pi*x))*sin(x-pi/4-n*pi/2)
end;

begin
done:=false;
writeln;
repeat
    write('Order? ');
    readln(ordr);
    if ordr<0.0 then done:=true
    else
        begin
            repeat
                write('Arg? ');
                readln(x)
            until x>=0.0;
            writeln('Y Bessel is ',bessy(x,ordr))
        end
    until done
end.

```



## ПРИЛОЖЕНИЕ Б. ПРОГРАММА НА ЯЗЫКЕ СИ

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

float bessy(float x, float n){
    const float small = 1.0E-8;
        const float euler =      0.57721566;
        const float pi = 3.1415926;
        const float pi2 = 0.63661977;

        float x2, sum, sum2, t, t2, ts, term, xx, y0, y1, ya, yb, yc, ans, a, b, sina, cosa;

    if(x<12){
        xx = 0.5 * x;
        x2 = xx * xx;
        t = log(xx) + euler;
        sum = 0.0;
        term = t;
        y0 = t;
        int j = 0;
        do{
            j = j+1;
            if(j != 1) sum = sum + 1/(j-1);
            ts = t-sum;
            term = -x2 * term / (j*j) * (1-1 / (j*ts));
            y0 = y0+term;
        }while(!(abs(term) < small));
        term = xx * (t-0.5);
        sum = 0.0;
        y1 = term;
        j = 1;
        do{
            j = j+1;
            sum = sum+1/(j-1);
            ts = t-sum;
            term = (-x2 * term) / (j * (j-1)) * ((ts-0.5 / j) / (ts + 0.5 / (j-1)));
            y1 = y1+term;
        }while(!(abs(term) < small));
        y0 = pi2 * y0;
        y1 = pi2 * (y1 - 1/x);
        if(n == 0.0){
            ans = y0;
        }else if(n == 1.0){
            ans = y1;
        }
    }
```

```

    }else{
        ts = 2.0/x;
        ya = y0;
        yb = y1;
        for (int j=2; j<=trunc(n+0.01); j+=1) {
            yc = ts*(j-1)*yb-ya;
            ya = yb;
            yb = yc;
        }
        ans = yc;
    }
    return ans;
}
}

int main(){
    float x, ordr;
    int done = 0;
    printf("\n");
    do{
        printf("Order? \n");
        scanf("%f", &ordr);
        if(ordr < 0.0){
            done = 1;
        }else{
            do{
                printf("Arg? \n");
                scanf("%f", &x);
            }while(!(x >= 0.0));
            printf("Y Bessel is %f \n", bessy(x,ordr));
        }
    }while(!(done));
    return 0;
}

```

## ПРИЛОЖЕНИЕ В.

### ПРОГРАММА НА ЯЗЫКЕ АССЕМБЛЕР

```
.file      "main.c"
.text
.def      scanf; .scl      3;      .type      32;      .endef
.seh_proc      scanf

scanf:
    pushq   %rbp
    .seh_pushreg   %rbp
    pushq   %rbx
    .seh_pushreg   %rbx
    subq    $56, %rsp
    .seh_stackalloc 56
    leaq    48(%rsp), %rbp
    .seh_setframe   %rbp, 48
    .seh_endprologue
    movq    %rcx, 32(%rbp)
    movq    %rdx, 40(%rbp)
    movq    %r8, 48(%rbp)
    movq    %r9, 56(%rbp)
    leaq    40(%rbp), %rax
    movq    %rax, -16(%rbp)
    movq    -16(%rbp), %rbx
    movl    $0, %ecx
    movq    imp____acrt_iob_func(%rip), %rax
    call    *%rax
    movq    %rbx, %r8
    movq    32(%rbp), %rdx
    movq    %rax, %rcx
    call    mingw_vfscanf
    movl    %eax, -4(%rbp)
    movl    -4(%rbp), %eax
    addq    $56, %rsp
    popq    %rbx
    popq    %rbp
    ret
.seh_endproc
.def      printf; .scl      3;      .type      32;      .endef
.seh_proc      printf

printf:
    pushq   %rbp
    .seh_pushreg   %rbp
    pushq   %rbx
    .seh_pushreg   %rbx
    subq    $56, %rsp
    .seh_stackalloc 56
    leaq    48(%rsp), %rbp
    .seh_setframe   %rbp, 48
    .seh_endprologue
    movq    %rcx, 32(%rbp)
    movq    %rdx, 40(%rbp)
    movq    %r8, 48(%rbp)
    movq    %r9, 56(%rbp)
    leaq    40(%rbp), %rax
    movq    %rax, -16(%rbp)
    movq    -16(%rbp), %rbx
    movl    $1, %ecx
    movq    imp____acrt_iob_func(%rip), %rax
    call    *%rax
    movq    %rbx, %r8
```

```

movq 32(%rbp), %rdx
movq %rax, %rcx
call    mingw_vfprintf
movl    %eax, -4(%rbp)
movl    -4(%rbp), %eax
addq    $56, %rsp
popq    %rbx
popq    %rbp
ret
.seh_endproc
.globl  bessy
.def    bessy; .scl 2; .type 32; .endef
.seh_proc    bessy

```

bessy:

```

pushq    %rbp
.seh_pushreg    %rbp
addq    $-128, %rsp
.seh_stackalloc    128
leaq    112(%rsp), %rbp
.seh_setframe    %rbp, 112
movaps    %xmm6, 0(%rbp)
.seh_savexmm    %xmm6, 112
.seh_endprologue
movss    %xmm0, 32(%rbp)
movss    %xmm1, 40(%rbp)
movss    .LC0(%rip), %xmm0
movss    %xmm0, -44(%rbp)
movss    .LC1(%rip), %xmm0
movss    %xmm0, -48(%rbp)
movss    .LC2(%rip), %xmm0
movss    %xmm0, -52(%rbp)
movss    .LC3(%rip), %xmm0
movss    %xmm0, -56(%rbp)
movss    .LC4(%rip), %xmm0
comiss    32(%rbp), %xmm0
jbe    .L22
movss    32(%rbp), %xmm1
movss    .LC5(%rip), %xmm0
mulss    %xmm1, %xmm0
movss    %xmm0, -60(%rbp)
movss    -60(%rbp), %xmm0
mulss    %xmm0, %xmm0
movss    %xmm0, -64(%rbp)
pxor    %xmm5, %xmm5
cvtss2sd -60(%rbp), %xmm5
movq    %xmm5, %rax
movq    %rax, %xmm0
call    log
pxor    %xmm1, %xmm1
cvtss2sd -48(%rbp), %xmm1
addsd    %xmm1, %xmm0
cvtss2sd %xmm0, %xmm0
movss    %xmm0, -68(%rbp)
pxor    %xmm0, %xmm0
movss    %xmm0, -4(%rbp)
movss    -68(%rbp), %xmm0
movss    %xmm0, -8(%rbp)
movss    -68(%rbp), %xmm0
movss    %xmm0, -12(%rbp)
movl    $0, -36(%rbp)

```

.L9:

```

addl    $1, -36(%rbp)

```

```

cmpl    $1, -36(%rbp)
je      .L8
movl    -36(%rbp), %eax
leal    -1(%rax), %ecx
movl    $1, %eax
cld
idivl   %ecx
pxor    %xmm0, %xmm0
cvtsi2ssl %eax, %xmm0
movss   -4(%rbp), %xmm1
addss   %xmm1, %xmm0
movss   %xmm0, -4(%rbp)

```

.L8:

```

movss   -68(%rbp), %xmm0
subss   -4(%rbp), %xmm0
movss   %xmm0, -72(%rbp)
movss   -64(%rbp), %xmm0
movss   .LC7(%rip), %xmm1
xorps   %xmm1, %xmm0
mulss   -8(%rbp), %xmm0
movl    -36(%rbp), %eax
imull   %eax, %eax
pxor    %xmm2, %xmm2
cvtsi2ssl %eax, %xmm2
movaps   %xmm0, %xmm1
divss   %xmm2, %xmm1
pxor    %xmm0, %xmm0
cvtsi2ssl -36(%rbp), %xmm0
movaps   %xmm0, %xmm3
mulss   -72(%rbp), %xmm3
movss   .LC8(%rip), %xmm0
movaps   %xmm0, %xmm2
divss   %xmm3, %xmm2
movss   .LC8(%rip), %xmm0
subss   %xmm2, %xmm0
mulss   %xmm1, %xmm0
movss   %xmm0, -8(%rbp)
movss   -12(%rbp), %xmm0
addss   -8(%rbp), %xmm0
movss   %xmm0, -12(%rbp)
movss   -8(%rbp), %xmm0
cvtss2sil    %xmm0, %eax
movl    %eax, %edx
negl    %edx
cmovns  %edx, %eax
pxor    %xmm1, %xmm1
cvtsi2ssl %eax, %xmm1
movss   -44(%rbp), %xmm0
comiss  %xmm1, %xmm0
seta    %al
xorl    $1, %eax
testb   %al, %al
jne     .L9
pxor    %xmm1, %xmm1
cvtss2sd -60(%rbp), %xmm1
pxor    %xmm0, %xmm0
cvtss2sd -68(%rbp), %xmm0
movsd   .LC9(%rip), %xmm2
subsd   %xmm2, %xmm0
mulsd   %xmm1, %xmm0
cvtsd2ss %xmm0, %xmm0
movss   %xmm0, -8(%rbp)

```

```

pxor    %xmm0, %xmm0
movss   %xmm0, -4(%rbp)
movss   -8(%rbp), %xmm0
movss   %xmm0, -16(%rbp)
movl    $1, -36(%rbp)

```

.L10:

```

addl    $1, -36(%rbp)
movl    -36(%rbp), %eax
leal    -1(%rax), %ecx
movl    $1, %eax
cld
idivl   %ecx
pxor    %xmm0, %xmm0
cvtsi2ssl %eax, %xmm0
movss   -4(%rbp), %xmm1
addss   %xmm1, %xmm0
movss   %xmm0, -4(%rbp)
movss   -68(%rbp), %xmm0
subss   -4(%rbp), %xmm0
movss   %xmm0, -72(%rbp)
movss   -64(%rbp), %xmm0
movss   .LC7(%rip), %xmm1
xorps   %xmm1, %xmm0
mulss   -8(%rbp), %xmm0
movl    -36(%rbp), %eax
subl    $1, %eax
imull   -36(%rbp), %eax
pxor    %xmm1, %xmm1
cvtsi2ssl %eax, %xmm1
divss   %xmm1, %xmm0
pxor    %xmm2, %xmm2
cvtss2sd %xmm0, %xmm2
pxor    %xmm0, %xmm0
cvtss2sd -72(%rbp), %xmm0
pxor    %xmm3, %xmm3
cvtsi2sdl -36(%rbp), %xmm3
movsd   .LC9(%rip), %xmm1
divsd   %xmm3, %xmm1
subsd   %xmm1, %xmm0
pxor    %xmm3, %xmm3
cvtss2sd -72(%rbp), %xmm3
movl    -36(%rbp), %eax
subl    $1, %eax
pxor    %xmm4, %xmm4
cvtsi2sdl %eax, %xmm4
movsd   .LC9(%rip), %xmm1
divsd   %xmm4, %xmm1
addsd   %xmm3, %xmm1
divsd   %xmm1, %xmm0
mulsd   %xmm2, %xmm0
cvtss2sd %xmm0, %xmm0
movss   %xmm0, -8(%rbp)
movss   -16(%rbp), %xmm0
addss   -8(%rbp), %xmm0
movss   %xmm0, -16(%rbp)
movss   -8(%rbp), %xmm0
cvtss2sil %xmm0, %eax
movl    %eax, %edx
negl    %edx
cmovns  %edx, %eax
pxor    %xmm1, %xmm1
cvtsi2ssl %eax, %xmm1

```

```

movss  -44(%rbp), %xmm0
comiss %xmm1, %xmm0
seta    %al
xorl    $1, %eax
testb   %al, %al
jne     .L10
movss   -12(%rbp), %xmm0
mulss   -56(%rbp), %xmm0
movss   %xmm0, -12(%rbp)
movss   .LC8(%rip), %xmm0
movaps  %xmm0, %xmm1
divss   32(%rbp), %xmm1
movss   -16(%rbp), %xmm0
subss   %xmm1, %xmm0
movss   -56(%rbp), %xmm1
mulss   %xmm1, %xmm0
movss   %xmm0, -16(%rbp)
pxor    %xmm0, %xmm0
ucomiss 40(%rbp), %xmm0
jp      .L11
pxor    %xmm0, %xmm0
ucomiss 40(%rbp), %xmm0
jne     .L11
movss   -12(%rbp), %xmm0
movss   %xmm0, -32(%rbp)
jmp     .L13

```

.L11:

```

movss   .LC8(%rip), %xmm0
ucomiss 40(%rbp), %xmm0
jp      .L14
movss   .LC8(%rip), %xmm0
ucomiss 40(%rbp), %xmm0
jne     .L14
movss   -16(%rbp), %xmm0
movss   %xmm0, -32(%rbp)
jmp     .L13

```

.L14:

```

movss   .LC10(%rip), %xmm0
divss   32(%rbp), %xmm0
movss   %xmm0, -72(%rbp)
movss   -12(%rbp), %xmm0
movss   %xmm0, -20(%rbp)
movss   -16(%rbp), %xmm0
movss   %xmm0, -24(%rbp)
movl    $2, -40(%rbp)
jmp     .L16

```

.L17:

```

movl    -40(%rbp), %eax
subl    $1, %eax
pxor    %xmm0, %xmm0
cvtsi2ssl %eax, %xmm0
mulss   -72(%rbp), %xmm0
mulss   -24(%rbp), %xmm0
subss   -20(%rbp), %xmm0
movss   %xmm0, -28(%rbp)
movss   -24(%rbp), %xmm0
movss   %xmm0, -20(%rbp)
movss   -28(%rbp), %xmm0
movss   %xmm0, -24(%rbp)
addl    $1, -40(%rbp)

```

.L16:

```

pxor    %xmm6, %xmm6

```

```

    cvtsi2sdl-40(%rbp), %xmm6
    pxor    %xmm1, %xmm1
    cvtss2sd 40(%rbp), %xmm1
    movsd   .LC11(%rip), %xmm0
    addsd   %xmm0, %xmm1
    movq    %xmm1, %rax
    movq    %rax, %xmm0
    call    trunc
    movq    %xmm0, %rax
    movq    %rax, %xmm4
    comisd  %xmm6, %xmm4
    jnb     .L17
    movss   -28(%rbp), %xmm0
    movss   %xmm0, -32(%rbp)

.L13:
    movss   -32(%rbp), %xmm0
    jmp     .L18

.L22:
    movss   -52(%rbp), %xmm0
    movaps  %xmm0, %xmm1
    mulss   32(%rbp), %xmm1
    movss   .LC10(%rip), %xmm0
    divss   %xmm1, %xmm0
    pxor    %xmm5, %xmm5
    cvtss2sd %xmm0, %xmm5
    movq    %xmm5, %rax
    movq    %rax, %xmm0
    call    sqrt
    movapd  %xmm0, %xmm6
    movss   -52(%rbp), %xmm0
    movss   .LC12(%rip), %xmm2
    movaps  %xmm0, %xmm1
    divss   %xmm2, %xmm1
    movss   32(%rbp), %xmm0
    subss   %xmm1, %xmm0
    movss   40(%rbp), %xmm1
    mulss   -52(%rbp), %xmm1
    movss   .LC10(%rip), %xmm2
    divss   %xmm2, %xmm1
    subss   %xmm1, %xmm0
    pxor    %xmm4, %xmm4
    cvtss2sd %xmm0, %xmm4
    movq    %xmm4, %rax
    movq    %rax, %xmm0
    call    sin
    mulsd   %xmm6, %xmm0
    cvtsd2ss %xmm0, %xmm0

.L18:
    movaps  0(%rbp), %xmm6
    subq    $-128, %rsp
    popq    %rbp
    ret
.seh_endproc
.def      main; .scl      2;      .type    32;      .endef
.section .rdata,"dr"

.LC13:
.ascii "\12\0"

.LC14:
.ascii "Order? \12\0"

.LC15:
.ascii "%f\0"

.LC16:

```



```

        .ascii "Arg? \12\0"
.LC17:
        .ascii "Y Bessel is %f \12\0"
        .text
        .globl  main
        .def    main;      .scl    2;      .type    32;      .endef
        .seh_proc      main

main:
        pushq   %rbp
        .seh_pushreg   %rbp
        movq    %rsp, %rbp
        .seh_setframe   %rbp, 0
        subq    $48, %rsp
        .seh_stackalloc 48
        .seh_endprologue
        call    main
        movl    $0, -4(%rbp)
        leaq    .LC13(%rip), %rax
        movq    %rax, %rcx
        call    printf

.L27:
        leaq    .LC14(%rip), %rax
        movq    %rax, %rcx
        call    printf
        leaq    -12(%rbp), %rax
        movq    %rax, %rdx
        leaq    .LC15(%rip), %rax
        movq    %rax, %rcx
        call    scanf
        movss   -12(%rbp), %xmm1
        pxor    %xmm0, %xmm0
        comiss   %xmm1, %xmm0
        ja      .L29
        jmp     .L24

.L29:
        movl    $1, -4(%rbp)
        jmp     .L26

.L24:
        leaq    .LC16(%rip), %rax
        movq    %rax, %rcx
        call    printf
        leaq    -8(%rbp), %rax
        movq    %rax, %rdx
        leaq    .LC15(%rip), %rax
        movq    %rax, %rcx
        call    scanf
        movss   -8(%rbp), %xmm0
        pxor    %xmm1, %xmm1
        comiss   %xmm1, %xmm0
        setnb   %al
        xorl    $1, %eax
        testb   %al, %al
        jne     .L24
        movss   -12(%rbp), %xmm0
        movl    -8(%rbp), %eax
        movaps   %xmm0, %xmm1
        movd    %eax, %xmm0
        call    bessy
        cvtss2sd %xmm0, %xmm0
        movq    %xmm0, %rax
        movq    %rax, %rdx
        movq    %rdx, %xmm0

```

```

    movapd %xmm0, %xmm1
    movq   %rax, %rdx
    leaq   .LC17(%rip), %rax
    movq   %rax, %rcx
    call   printf
.L26:
    cmpl   $0, -4(%rbp)
    je     .L27
    movl   $0, %eax
    addq   $48, %rsp
    popq   %rbp
    ret
.seh_endproc
.section .rdata,"dr"
.align 4
.LC0:
    .long  841731191
    .align 4
.LC1:
    .long  1058260072
    .align 4
.LC2:
    .long  1078530010
    .align 4
.LC3:
    .long  1059256707
    .align 4
.LC4:
    .long  1094713344
    .align 4
.LC5:
    .long  1056964608
    .align 16
.LC7:
    .long  -2147483648
    .long  0
    .long  0
    .long  0
    .align 4
.LC8:
    .long  1065353216
    .align 8
.LC9:
    .long  0
    .long  1071644672
    .align 4
.LC10:
    .long  1073741824
    .align 8
.LC11:
    .long  1202590843
    .long  1065646817
    .align 4
.LC12:
    .long  1082130432
    .ident "GCC: (GNU) 11.2.0"
    .def   mingw_vfscanf;    .scl    2;    .type   32;    .endef
    .def   mingw_vfprintf;   .scl    2;    .type   32;    .endef
    .def   log;              .scl    2;    .type   32;    .endef
    .def   trunc;            .scl    2;    .type   32;    .endef
    .def   sqrt;             .scl    2;    .type   32;    .endef
    .def   sin;              .scl    2;    .type   32;    .endef

```

