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

Студентка гр. 7304 _____ Юруть Е.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы

Изучение и сравнение метрик Холстеда для программ на C, Pascal и ассемблере.

Постановка задачи

Для заданного варианта программы обработки данных, представленной на языке Паскаль, разработать вычислительный алгоритм и также 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 "Code generation/Generate assembler source» 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.

В заданных на Паскале вариантах программ обработки данных важен только вычислительный алгоритм, реализуемый программой. Поэтому для получения более корректных оценок характеристик программ следует учитывать только вычислительные операторы и исключить операторы, обеспечивающие интерфейс с пользователем и выдачу текстовых сообщений.

В сути алгоритма, реализуемого программой, нужно разобраться достаточно хорошо для возможности внесения в программу модификаций, выполняемых в дальнейшем при проведении измерений и улучшении характеристик качества программы.

Для измеряемых версий программ в дальнейшем будет нужно исключить операции ввода данных с клавиатуры и вывода на печать, потребляющие основную долю ресурса времени при выполнении программы.

Поэтому можно уже в этой работе предусмотреть соответствующие преобразования исходной программы.

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 характеристик длина программы, уровень программы, время программирования 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. Для программы на Ассемблере возможен только ручной расчет характеристик. При ручном расчете, в отличие от программного, нужно учитывать только выполняемые операторы, а все описания не учитываются. Соответственно все символы («;», «=», переменные, цифры), входящие в описания, не учитываются.
2) с помощью программы автоматизации расчета метрик Холстеда (для Си- 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.

Результаты расчетов представить в виде таблиц с текстовыми комментариями:

1) Паскаль. Ручной расчет:
   а) Измеримые характеристики, б) Расчетные характеристики
2) Паскаль. Программный расчет:
   а) Измеримые характеристики, б) Расчетные характеристики
3) Си. Ручной расчет:
   а) Измеримые характеристики, б) Расчетные характеристики
4) Си. Программный расчет:
   а) Измеримые характеристики, б) Расчетные характеристики
5) Ассемблер. Ручной расчет:
   а) Измеримые характеристики, б) Расчетные характеристики
6) Сводная таблица расчетов для трех языков.

Ход работы

1. Исходный код программы на языке программирования Pascal представлен в Приложении А.

2. Написана программа на языке программирования Си, реализующая заданный алгоритм. Исходный код в Приложении В.

3. Получен код программы на языке Assembler в результате компиляции текста программы, написанной на языке Си. Исходный код в Приложении С.

4. Расчет характеристик:

1) Ручной расчет для программы на Pascal.

а) Ручной расчёт измеримых характеристик представлен в Таблице 1.

Оператор	**I**	**F_{1i}**	**Операнд**	**J**	**F_{2j}**	**Операнд**	**J**	**F_{2j}**
;	1	15	done	1	4	0.0	30	3
:=	2	16	ec	2	6	0.01693122	31	1
() или begin.. end	3	32	er	3	6	0.07619048	32	1
*	4	31	x	4	16	1	33	13
+	5	23	x2	5	18	1.0	34	5
-	6	10	erfc	6	1	1.5	35	1
/	7	14	i	7	1	1.7724538	36	2
<	8	2	sqrtpi	8	4	10	37	1
=	9	1	sum	9	8	11	38	1
If…then..else	10	3	t10	10	2	12	39	3
repeat	11	1	t11	11	2	2	40	1
erf	12	1	t12	12	2	2.0	41	2
erfc	13	1	t2	13	2	3	42	1
exp	14	2	t3	14	2	3.07843E-3	43	1
			t4	15	2	4	44	2
			t5	16	2	4.736005E-4	45	1
			t6	17	2	5	46	1
			t7	18	2	6	47	1
			t8	19	2	6.314673E-5	48	1
			t9	20	2	6.476214E-9	49	1
			v	21	14	7	50	1
			true	22	1			
			false	23	1			
			erf	24	1			

			7.429027E-6	25	1			
			7.447646E-8	26	1			
			7.820028E-7	27	1			
			8	28	2			
			9	29	1			
Сумма		**152**						**153**

Таблица 1. Ручной расчет измеримых характеристик Pascal

b) Ручной расчет расчетных характеристик представлен в Таблице 2.

Характеристика	**Значение**
Число простых операторов η_1	14
Число простых операндов η_2	50
Общее число всех операторов N_1	152
Общее число всех операндов N_2	153
Словарь η	64
Длина N	305
Теоретическая длина $\hat{N}$	335.50
Объём V	1830.00
Потенциальный объём V*	11.61
Уровень программы L	0.00634
Оценка уровня программы $\tilde{L}$	0.04669
Интеллектуальное содержание I	85.43
Работа программирования E	288459.55
Оценка времени программирования $\hat{T}$	3919.86
Время программирования T	28845.96
Уровень языка λ	0.074
Ожидаемое число ошибок в программе B	2

Таблица 2. Ручной расчет расчетных характеристик Pascal

2) Программный расчет для программы на Pascal.

a) Программный расчёт измеримых характеристик представлен в Таблице 3.

Оператор	**I**	**F_{1i}**	**Операнд**	**J**	**F_{2j}**	**Операнд**	**J**	**F_{3j}**	**Операнд**	**J**	**F_{3j}**
()	1	37	‘, Erf=’	1	1	7.447646 E-8	27	1	v	53	14
*	2	31	‘, Erfc=’	2	1	7.820028	28	1	x	54	16

						E-7					
+	3	23	‘Arg? ’	3	1	8	29	2	x2	55	18
-	4	3	‘X= ’	4	1	9	30	1	0.666666 67	56	2
/	5	14	0.0	5	3	done	31	4			
;	6	48	0.01693122	6	1	ec	32	6			
<	7	2	0.07619048	7	1	er	33	6			
=	8	30	1	8	13	erf	34	1			
ClrScr	9	1	1.0	9	5	erfc	35	1			
boolean	10	1	1.5	10	1	erfd4	36	1			
const	11	2	1.7724538	11	2	false	37	1			
erf	12	2	10	12	1	i	38	1			
erfc	13	2	11	13	1	sqrtpi	39	4			
exp	14	2	12	14	3	sum	40	8			
function	15	2	2	15	1	t10	41	2			
if	16	3	2.0	16	2	t11	42	2			
integer	17	1	3	17	1	t12	43	2			
program	18	1	3.07843E-3	18	1	t2	44	2			
readln	19	1	4	19	2	t3	45	2			
real	20	7	4.736005E-4	20	1	t4	46	2			
repeat	21	1	5	21	1	t5	47	2			
write	22	1	6	22	1	t6	48	2			
writeln	23	2	6.314673E-5	23	2	t7	49	2			
			6.476214E-9	24	1	t8	50	2			
			7	25	1	t9	51	2			
			7.429027E-6	26	1	true	52	1			
Сумма		**217**									**160**

Таблица 3. Программный расчет измеримых характеристик Pascal

b) Программный расчёт расчетных характеристик представлен в Таблице 4.

Характеристика	**Значение**
Число простых операторов η_1	23
Число простых операндов η_2	56
Общее число всех операторов N_1	217

Общее число всех операндов N_2	160
Словарь η	79
Длина N	377
Теоретическая длина $\hat{N}$	429,254
Объём V	2476,53
Потенциальный объём V*	11,61
Уровень программы L	0,008269
Оценка уровня программы $\tilde{L}$	0,0304348
Интеллектуальное содержание I	72,329
Работа программирования E	287402
Оценка времени программирования $\hat{T}$	4939,38
Время программирования T	15966,8
Уровень языка λ	0,162498
Ожидаемое число ошибок в программе B	1,45168

Таблица 4. Программный расчет расчетных характеристик Pascal

Ручной расчет для программы на C.

а) Ручной расчёт измеримых характеристик представлен в Таблице 5.

Оператор	**I**	**F_{1i}**	**Операнд**	**J**	**F_{2j}**	**Операнд**	**J**	**F_{2j}**
()	1	25	done	1	4	0.0	27	3
*	2	47	ec	2	6	0.01693122	28	1
+	3	23	er	3	6	0.07619048	29	1
,	4	11	i	4	1	1	30	13
-	5	10	sqrtpi	5	4	1.0	31	5
/	6	30	sum	6	8	1.5	32	1
;	7	22	t10	7	2	1.7724538	33	2
<	8	5	t11	8	2	10	34	1
=	9	30	t12	9	2	11	35	1
==	10	1	t2	10	2	12	36	3
erf	11	2	t3	11	2	2	37	1
erfc	12	2	t4	12	2	2.0	38	2
do…while	13	1	t5	13	2	3	39	1
exp	14	2	t6	14	2	3.07843E-3	40	1
if…else	15	3	t7	15	2	4	41	2
			t8	16	2	4.736005E-4	42	1
			t9	17	2	5	43	1

			v	18	14	6	44	1
			x	19	16	6.314673E-5	45	1
			x2	20	18			
			7.429027E-6	21	1			
			7.447646E-8	22	1			
			7.820028E-7	23	1			
			8	24	2			
			9	25	1			
			6.476214E-9	26	1			
Сумма		**214**						**148**

Таблица 5. Ручной расчет измеримых характеристик C

b) Ручной расчет расчетных характеристик представлен в Таблице 6.

Характеристика	**Значение**
Число простых операторов η_1	15
Число простых операндов η_2	45
Общее число всех операторов N_1	214
Общее число всех операндов N2	148
Словарь η	60
Длина N	362
Теоретическая длина $\hat{N}$	305.74
Объём V	2138.29
Потенциальный объём V*	11,61
Уровень программы L	0.00543
Оценка уровня программы $\tilde{L}$	0.04054
Интеллектуальное содержание I	86.69
Работа программирования E	393838.11
Оценка времени программирования $\hat{T}$	5274.46
Время программирования T	39383.81

Уровень языка λ	0.063
Ожидаемое число ошибок в программе B	3

Таблица 6. Ручной расчет расчетных характеристик C

3) Программный расчет для программы на C.

а) Программный расчёт измеримых характеристик представлен в Таблице7.

Оператор	**I**	**F_{1i}**	**Операнд**	**J**	**F_{2j}**	**Операнд**	**J**	**F_{2j}**
()	1	1	“%lf”	1	8	7,447646E-8	30	1
*	2	5	“Arg? ”	2	14	7,820028E-7	31	1
+	3	32	“X= %8.4f, Erf= %12f, Erfc = %12f\n”	3	1	8	32	1
,	4	1	“\n”	4	2	9	33	1
-	5	50	“clr”	5	1	done	34	4
/	6	9	0	6	1	ec	35	6
;	7	26	0,0	7	3	er	36	6
<	8	6	0,01693122	8	3	i	37	1
=	9	2	0,07619048	9	30	sqrtpi	38	4
==	10	42	0,6666667	10	19	sum	39	8
_&	11	5	1	11	9	t10	40	2
_-	12	2	1,0	12	9	t11	41	2
const	13	1	1,5	13	6	t12	42	2
double	14	6	1,7724538	14	3	t2	43	2
dowhile	15	4	10	15	9	t3	44	2
erf	16	2	11	16	3	t4	45	2
erfc	17	2	12	17	10	t5	46	2
exp	18	1	2	18	22	t6	47	2
if	19	1	2,0	19	25	t7	48	2
int	20	1	3	20	1	t8	49	2
main	21	4	3,078403E-3	21	1	t9	50	2
printf	22	2	4	22	1	v	51	14
return	23	5	4,736005E-4	23	1	x	52	16
scanf	24	1	5	24	1	x2	53	18
system	25	4	6	25	1			
			6,314673E-5	26	1			
			6,476214E-9	27	1			

			7	28	1			
			7,429027E-6	29	1			
Сумма		**239**						**154**

Таблица 7. Программный расчет измеримых характеристик С

b) Программный расчёт расчетных характеристик представлен в Таблице 8.

Характеристика	**Значение**
Число простых операторов η_1	25
Число простых операндов η_2	53
Общее число всех операторов N_1	239
Общее число всех операндов N_2	154
Словарь η	78
Длина N	393
Теоретическая длина $\hat{N}$	419,676
Объём V	2470,16
Потенциальный объём V*	11,61
Уровень программы L	0.00795554
Оценка уровня программы $\tilde{L}$	0.0275325
Интеллектуальное содержание I	68.0097
Работа программирования E	310496
Оценка времени программирования $\hat{T}$	5322,67
Время программирования T	17249,8
Уровень языка λ	0,156338
Ожидаемое число ошибок в программе B	1,52844

Таблица 8. Программный расчет расчетных характеристик С

4) Ручной расчет для программы на Assembler.

a) Ручной расчёт измеримых характеристик представлен в Таблице 9.

Оператор	**I**	**F_{1i}**	**Операнд**	**J**	**F_{2j}**
Pushl	1	4	$152	1	1
movl	2	18	$72	2	1

subl	3	3	$-16	3	1
fldl	4	56	$LC23	4	1
fstpl	5	32	$1	5	1
fmul	6	2	$10	6	1
faddl	7	9	$LC24	7	1
faddp	8	15	$0	8	2
fchs	9	1	$LC28	9	1
Jp L21	10	1	$_x	10	1
fadd	11	2	%ebp	11	6
fdivl	12	1	8(%ebp)	12	2
fmull	13	14	-128(%ebp)	13	3
fmulp	14	12	12(%ebp)	14	2
fld1	15	18	-112(%ebp)	15	13
leave	16	3	-16(%ebp)	16	4
ret	17	3	-24(%ebp)	17	5
fdivp	18	5	-32(%ebp)	18	15
fdivrp	19	5	-40(%ebp)	19	6
fld	20	1	-48(%ebp)	20	5
ret	21	3	-56(%ebp)	21	2
andl	22	1	-64(%ebp)	22	2
subl	23	3	-72(%ebp)	23	2
movb	24	2	-80(%ebp)	24	2
fldz	25	4	-88(%ebp)	25	2
fcommp	26	2	-96(%ebp)	26	2
fnstsw	27	4	-104(%ebp)	27	2
sahf	28	4	-128(%ebp)	28	3
Jbe L19	29	1	-120(%ebp)	29	2
Jmp L8	30	2	%esp	30	16
Jbe L20	31	1	20(%esp)	31	1
Jmp L13	32	1	12(%esp)	32	1
Call _exp	33	2	4(%esp)	33	2
Call__main	34	1	%eax	34	10
fsubp	43	2	%st(0)	35	6
movezbl	44	1	%st(1)	36	43
testb	45	1	%st	37	42

Call _erf	38	1	%st(2)	38	1
Call _erfc	39	1	%al	39	2
fucomp	40	2	%ax	40	4
Jne L9	41	1	$32	41	1
Jne L14	42	1			
Сумма		**246**			**220**

Таблица 9. Ручной расчет измеримых характеристик Assembler

b) Ручной расчет расчетных характеристик представлен в Таблице 10.

Характеристика	**Значение**
Число простых операторов η_1	42
Число простых операндов η_2	41
Общее число всех операторов N_1	246
Общее число всех операндов N_2	220
Словарь η	83
Длина N	466
Теоретическая длина $\hat{N}$	446.14
Объём V	2970.77
Потенциальный объём V*	11.61
Уровень программы L	0.00391
Оценка уровня программы $\tilde{L}$	0.00887
Интеллектуальное содержание I	26.36
Работа программирования E	760186.81
Оценка времени программирования $\hat{T}$	33475.49
Время программирования T	76018.68
Уровень языка λ	0.045

Ожидаемое число ошибок в программе B	3

Таблица 10. Ручной расчет расчетных характеристик Assembler

5) Сводная таблица расчетов для трех языков

Метрика/Язык	Pascal		C		Assembler
	Ручной	Программный	Ручной	Программный	Ручной
$\lvert_1$	14	23	15	25	42
$\lvert_2$	50	56	45	53	41
N_1	152	217	214	239	246
N_2	153	160	148	154	220
$\lvert$	64	79	60	78	83
N	305	377	362	393	466
$\dot{N}$	335.50	429,254	305.74	419,676	446.14
V	1830.00	2476,53	2138.29	2470,16	2970.77
V*	11.61	11,61	11,61	11,61	11.61
L	0.00634	0,008269	0.00543	0.00795554	0.00391
$\acute{L}$	0.04669	0,0304348	0.04054	0.0275325	0.00887
I	85.43	72,329	86.69	68.0097	26.36
E	288459.55	287402	393838.11	310496	760186.81
$\check{T}$	3919.86	4939,38	5274.46	5322,67	33475.49
T	28845.96	15966,8	39383.81	17249,8	76018.68
λ	0.074	0,162498	0.063	0,156338	0.045
B	2	1,45168	3	1,52844	3

Таблица 11. Сводная таблица расчетов для языков Pascal, C, Assembler

Вывод

В результате выполнения данной лабораторной работы была изучена система метрик Холстеда. Было проведено сравнение программ на языках Pascal, Си и Ассемблер

Приложение А. Исходный код программы на Pascal

```
program erfd4;
uses Crt;
{ evaluation of the gaussian error function }

var     x,er,ec         : real;
        done            : boolean;

function erf(x: real): real;
{ infinite series expansion of the Gaussian error function }

const   sqrtpi          = 1.7724538;
        t2              = 0.66666667;
        t3              = 0.66666667;
        t4              = 0.07619048;
        t5              = 0.01693122;
        t6              = 3.078403E-3;
        t7              = 4.736005E-4;
        t8              = 6.314673E-5;
        t9              = 7.429027E-6;
        t10             = 7.820028E-7;
        t11             = 7.447646E-8;
        t12             = 6.476214E-9;

var     x2,sum          : real;
        i               : integer;

begin
        x2:=x*x;
        sum:=t5+x2*(t6+x2*(t7+x2*(t8+x2*(t9+x2*(t10+x2*(t11+x2*t12))))));
        erf:=2.0*exp(-x2)/sqrtpi*(x*(1+x2*(t2+x2*(t3+x2*(t4+x2*sum)))))
end;    { function erf }

function erfc(x: real): real;
{ complement of error function }
const   sqrtpi          = 1.7724538;

var     x2,v,sum        : real;

begin
        x2:=x*x;
        v:=1.0/(2.0*x2);
        sum:=v/(1+8*v/(1+9*v/(1+10*v/(1+11*v/(1+12*v)))));
        sum:=v/(1+3*v/(1+4*v/(1+5*v/(1+6*v/(1+7*sum)))));
```

```
        erfc:=1.0/(exp(x2)*x*sqrtpi*(1+v/(1+2*sum)))
end;            { function ercf }

begin           { main }
                done:=false;
        {writeln;}

                {write('Arg? ');}
                {readln(x);}
                if x<0.0 then done:=true
                else
                        begin
                                if x=0.0 then
                                        begin
                                                er:=0.0;
                                                ec:=1.0
                                        end
                                else
                                        begin
                                                if x<1.5 then
                                                        begin
                                                                er:=erf(x);
                                                                ec:=1.0-er
                                                        end
                                                else
                                                        begin
                                                                ec:=erfc(x);
                                                                er:=1.0-ec
                                                        end     { if }
                                        end;
                                {writeln('X= ',x:8:4,', Erf= ',er:12,', Erfc= ',ec:12)}
                        end     { if }
   until done
end.
```

Приложение B. Исходный код программы на C

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

/* evaluation of the gaussian error function */

double x, er, ec;
bool done;
```

```
double erf(double x)
{
   const double sqrtpi = 1.7724538;
   const double t2 = 0.66666667;
   const double t3 = 0.66666667;
   const double t4 = 0.07619048;
   const double t5 = 0.01693122;
   const double t6 = 3.078403E-3;
   const double t7 = 4.736005E-4;
   const double t8 = 6.314673E-5;
   const double t9 = 7.429027E-6;
   const double t10 = 7.820028E-7;
   const double t11 = 7.447646E-8;
   const double t12 = 6.476214E-9;

   double x2, sum;
   int i;

   x2 = x * x;
   sum = t5 + x2 * (t6 + x2 * (t7 + x2 * (t8 + x2 * (t9 + x2 * (t10 + x2 * (t11 + x2 * t12))))));

   return 2.0 * exp(-x2) / sqrtpi * (x * (1 + x2 * (t2 + x2 * (t3 + x2 * (t4 + x2 * sum)))));
} /* function erf */

double erfc(double x)
{
   /* complement of error function */

   const double sqrtpi = 1.7724538;
   double x2, v, sum;

   x2 = x * x;
   v = 1.0 / (2.0 * x2);
   sum = v / (1 + 8 * v / (1 + 9 * v / (1 + 10 * v / (1 + 11 * v / (1 + 12 * v)))));
   sum = v / (1 + 3 * v / (1 + 4 * v / (1 + 5 * v / (1 + 6 * v / (1 + 7 * sum)))));
   return 1.0 / (exp(x2) * x * sqrtpi * (1 + v / (1 + 2 * sum)));
} /* function ercf */

int main() /* main */
{
  /* system("clr");*/
   done = true;
  /* printf("\n");*/
   do {
```

```
        printf("Arg? ");
        scanf("%lf", &x);
        if (x < 0.0){
           done = false;
        }
        else if (x == 0.0)
        {
           er = 0.0;
           ec = 1.0;
        }
        else
        {
           if (x < 1.5)
           {

              er = erf(x);
              ec = 1.0 - er;
           }
           else
           {
              ec = erfc(x);
              er = 1.0 - ec;
              /* if */
           }
          /* printf("X= %8.4f, Erf= %12f, Erfc= %12f\n", x, er, ec);*/
        } /* if */
    } while (done);
}
/* End. */
```

Приложение C. Исходный код программы на Assembler

```
                .file "lab1.c"
.text
.comm _x, 8, 3
.comm _er, 8, 3
.comm _ec, 8, 3
.comm _done, 1, 0
.globl _erf
.def _erf; .scl 2; .type 32; .endef
_erf:
LFB13:
.cfi_startproc
pushl %ebp
.cfi_def_cfa_offset 8
.cfi_offset 5, -8
movl %esp, %ebp
```

```
.cfi_def_cfa_register 5
subl $152, %esp
movl 8(%ebp), %eax
movl %eax, -128(%ebp)
movl 12(%ebp), %eax
movl %eax, -124(%ebp)
fldl LC0
fstpl -16(%ebp)
fldl LC1
fstpl -24(%ebp)
fldl LC1
fstpl -32(%ebp)
fldl LC2
fstpl -40(%ebp)
fldl LC3
fstpl -48(%ebp)
fldl LC4
fstpl -56(%ebp)
fldl LC5
fstpl -64(%ebp)
fldl LC6
fstpl -72(%ebp)
fldl LC7
fstpl -80(%ebp)
fldl LC8
fstpl -88(%ebp)
fldl LC9
fstpl -96(%ebp)
fldl LC10
fstpl -104(%ebp)
fldl -128(%ebp)
fmul %st(0), %st
fstpl -112(%ebp)
fldl -112(%ebp)
fmull -104(%ebp)
faddl -96(%ebp)
fmull -112(%ebp)
faddl -88(%ebp)
fmull -112(%ebp)
faddl -80(%ebp)
fmull -112(%ebp)
faddl -72(%ebp)
fmull -112(%ebp)
faddl -64(%ebp)
fmull -112(%ebp)
```

```
faddl -56(%ebp)
fmull -112(%ebp)
fldl -48(%ebp)
faddp %st, %st(1)
fstpl -120(%ebp)
fldl -112(%ebp)
fchs
fstpl (%esp)
call _exp
fadd %st(0), %st
fdivl -16(%ebp)
fldl -112(%ebp)
fmull -120(%ebp)
faddl -40(%ebp)
fmull -112(%ebp)
faddl -32(%ebp)
fmull -112(%ebp)
faddl -24(%ebp)
fmull -112(%ebp)
fld1
faddp %st, %st(1)
fmull  -128(%ebp)
fmulp %st, %st(1)
leave
.cfi_restore 5
.cfi_def_cfa 4, 4
ret
.cfi_endproc
LFE13:
.globl _erfc
.def _erfc; .scl 2; .type 32; .endef
_erfc:
LFB14:
.cfi_startproc
pushl %ebp
.cfi_def_cfa_offset 8
.cfi_offset 5, -8
movl %esp, %ebp
.cfi_def_cfa_register 5
subl $72, %esp
movl 8(%ebp), %eax
movl %eax, -48(%ebp)
movl 12(%ebp), %eax
movl %eax, -44(%ebp)
fldl LC0
```

```
fstpl -16(%ebp)
fldl -48(%ebp)
fmul %st(0), %st
fstpl -24(%ebp)
fldl -24(%ebp)
fadd %st(0), %st
fld1
fdivp %st, %st(1)
fstpl -32(%ebp)
fldl -32(%ebp)
fldl LC13
fmulp %st, %st(1)
fldl -32(%ebp)
fldl LC14
fmulp %st, %st(1)
fldl -32(%ebp)
fldl LC15
fmulp %st, %st(1)
fldl -32(%ebp)
fldl LC16
fmulp %st, %st(1)
fldl -32(%ebp)
fldl LC17
fmulp %st, %st(1)
fld1
faddp %st, %st(1)
fdivrp %st, %st(1)
fld1
faddp %st, %st(1)
fdivrp %st, %st(1)
fld1
faddp %st, %st(1)
fdivrp %st, %st(1)
fld1
faddp %st, %st(1)
fdivrp %st, %st(1)
fld1
faddp %st, %st(1)
fldl -32(%ebp)
fdivp %st, %st(1)
fstpl -40(%ebp)
fldl -32(%ebp)
fldl LC18
fmulp %st, %st(1)
fldl -32(%ebp)
```

```
fldl LC19
fmulp %st, %st(1)
fldl -32(%ebp)
fldl LC20
fmulp %st, %st(1)
fldl -32(%ebp)
fldl LC21
fmulp %st, %st(1)
fldl -40(%ebp)
fldl LC22
fmulp %st, %st(1)
fld1
faddp %st, %st(1)
fdivrp %st, %st(1)
fld1
faddp %st, %st(1)
fdivrp %st, %st(1)
fld1
faddp %st, %st(1)
fdivrp %st, %st(1)
fld1
faddp %st, %st(1)
fdivrp %st, %st(1)
fld1
faddp %st, %st(1)
fldl -32(%ebp)
fdivp %st, %st(1)
fstpl -40(%ebp)
fldl -24(%ebp)
fstpl (%esp)
call _exp
fmull -48(%ebp)
fmull -16(%ebp)
fldl -40(%ebp)
fld %st(0)
faddp %st, %st(1)
fld1
faddp %st, %st(1)
fldl -32(%ebp)
fdivp %st, %st(1)
fld1
faddp %st, %st(1)
fmulp %st, %st(1)
fld1
fdivp %st, %st(1)
```

```
leave
.cfi_restore 5
.cfi_def_cfa 4, 4
ret
.cfi_endproc
LFE14:
.def___main; .scl 2; .type 32; .endef
.section .rdata,"dr"
LC23:
.ascii "clr\0"
LC24:
.ascii "Arg? \0"
LC25:
.ascii "%lf\0"
.align 4
LC28:
.ascii "X= %8.4f, Erf= %12f, Erfc= %12f\12\0"
.text
.globl _main
.def _main; .scl 2; .type 32; .endef
_main:
LFB15:
.cfi_startproc
pushl %ebp
.cfi_def_cfa_offset 8
.cfi_offset 5, -8
movl %esp, %ebp
.cfi_def_cfa_register 5
andl $-16, %esp
subl $32, %esp
call___main
movl $LC23, (%esp)
call _system
movb $1, _done
movl $10, (%esp)
call  _putchar
L14:
movl $LC24, (%esp)
call _printf
movl $_x, 4(%esp)
movl $LC25, (%esp)
call _scanf
fldl _x
fldz
fcompp
```

```
fnstsw %ax
sahf
jbe L19
movb $0, _done
jmp L8
L19:
fldl _x
fldz
fucomp %st(1)
fnstsw %ax

sahf
jp L21
fldz
fucompp
fnstsw %ax
sahf
jne L9
fldz
fstpl _er
fld1
fstpl _ec
jmp L8
L21:
fstp %st(0)
L9:
fldl _x
fldl LC27
fcompp
fnstsw %ax
sahf
jbe L20
fldl _x
fstpl (%esp)
call _erf
fstpl  _er
fldl _er
fld1
fsubp %st, %st(1)
fstpl _ec
jmp L13
L20:
fldl _x
fstpl (%esp)
call _erfc
```

```
fstpl _ec
fldl _ec
fld1
fsubp %st, %st(1)
fstpl _er
L13:
fldl _ec
fldl _er
fldl _x
fxch %st(2)
fstpl 20(%esp)
fstpl 12(%esp)
fstpl 4(%esp)
movl $LC28, (%esp)
call _printf
L8:
movzbl _done, %eax
testb %al, %al
jne L14
movl $0, %eax
leave
.cfi_restore 5
.cfi_def_cfa 4, 4
ret
.cfi_endproc
LFE15:
.section .rdata,"dr"
.align 8
LC0:
.long -2079671268
.long 1073503224
.align 8
LC1:
.long 1461679763
.long 1071994197
.align 8
LC2:
.long 601740724
.long 1068728632
.align 8
LC3:
.long -256263175
.long 1066489450
.align 8
LC4:
```

```
.long 112156783
.long 1063860193
.align 8
LC5:
.long -1465145423
.long 1061095858
.align 8
LC6:
.long -1886360939
.long 1058049460
.align 8
LC7:
.long 1685304208
.long 1054812379
.align 8
LC8:
.long 1473632397
.long 1051344218
.align 8
LC9:
.long -1115619875
.long 1047789051
.align 8
LC10:
.long 793809704
.long 1044107436
.align 8
LC13:
.long 0
.long 1075838976
.align 8
LC14:
.long 0
.long 1075970048
.align 8
LC15:
.long 0
.long 1076101120
.align 8
LC16:
.long 0
.long 1076232192
.align 8
LC17:
.long 0
```

```
.long 1076363264
.align 8
LC18:
.long 0
.long 1074266112
.align 8
LC19:
.long 0
.long 1074790400
.align 8
LC20:
.long 0
.long 1075052544
.align 8
LC21:
.long 0
.long 1075314688
.align 8
LC22:
.long 0
.long 1075576832
.align 8
LC27:
.long 0
.long 1073217536
.ident "GCC: (MinGW.org GCC Build-20200227-1) 9.2.0"
.def _exp; .scl 2; .type 32; .endef
.def _system; .scl 2; .type 32; .endef
.def _putchar; .scl 2; .type 32; .endef
.def _printf; .scl 2; .type 32; .endef
.def _scanf; .scl 2; .type 32; .endef
```