**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ОТЧЕТ**

**по лабораторной работе №1**

**по дисциплине «Качество и метрология программного обеспечения» Тема: «Расчет метрических характеристик качества разработки по метрикам Холстеда»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8304 |  | Сергеев А.Д. |
| Преподаватель |  | Кирьянчиков В.А. |

Санкт-Петербург

# 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 (c 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

Программы на языке Паскаль, С и языке ассемблера представлены в приложениях A, Б и В, соответственно. Программа на языке ассемблера была сгенерирована с помощью инструмента с сайта <https://gcc.godbolt.org>.

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

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

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Оператор** | **Количество** | **№** | **Операнд** | **Количество** | **№** | **Операнд** | **Количество** |
| 1 | () | 17 | 1 | ' Erf= ' | 1 | 21 | erfd3 | 1 |
| 2 | \* | 13 | 2 | ', Erfc= ' | 1 | 22 | false | 1 |
| 3 | + | 6 | 3 | 'Arg? ' | 1 | 23 | i | 7 |
| 4 | - | 4 | 4 | 'X= ' | 1 | 24 | sqrtpi | 4 |
| 5 | / | 5 | 5 | 0 | 1 | 25 | sum | 10 |
| 6 | < | 3 | 6 | 0.0 | 3 | 26 | sum1 | 3 |
| 7 | = | 28 | 7 | 1 | 2 | 27 | term | 6 |
| 8 | ClrScr | 1 | 8 | 1.0 | 8 | 28 | terms | 3 |
| 9 | const | 2 | 9 | 1.0E-4 | 1 | 29 | tol | 2 |
| 10 | erf | 2 | 10 | 1.5 | 1 | 30 | true | 1 |
| 11 | erfc | 2 | 11 | 1.7724538 | 2 | 31 | u | 4 |
| 12 | exp | 2 | 12 | 12 | 3 | 32 | v | 4 |
| 13 | for | 1 | 13 | 2.0 | 4 | 33 | x | 17 |
| 14 | if | 3 | 14 | 4 | 1 | 34 | X2 | 8 |
| 15 | program | 1 | 15 | 8 | 2 |  |  |  |
| 16 | readln | 1 | 16 | done | 4 |  |  |  |
| 17 | real | 2 | 17 | ec | 6 |  |  |  |
| 18 | repeat | 2 | 18 | er | 6 |  |  |  |
| 19 | write | 1 | 19 | erf | 1 |  |  |  |
| 20 | writeln | 2 | 20 | erfc | 1 |  |  |  |

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Оператор** | **Количество** | **№** | **Операнд** | **Количество** |
| 1 | () | 15 | 1 | "%lf" | 1 |
| 2 | \* | 13 | 2 | "Arg? " | 1 |
| 3 | + | 6 | 3 | "X = %.8lf; Erf = %.12lf; Erfc = %.12lf\n" | 1 |
| 4 | -- | 1 | 4 | 1 | 2 |
| 5 | , | 9 | 5 | 1.0 | 3 |
| 6 | - | 2 | 6 | 1.0E-4 | 3 |
| 7 | / | 5 | 7 | 1.5 | 8 |
| 8 | < | 3 | 8 | 1.7724538 | 1 |
| 9 | = | 25 | 9 | 12 | 1 |
| 10 | == | 1 | 10 | 2.0 | 1 |
| 11 | >= | 1 | 11 | 2 | 1 |
| 12 | \_& | 1 | 12 | 2.0 | 4 |
| 13 | \_- | 2 | 13 | done | 3 |
| 14 | dowhile | 3 | 14 | ec | 6 |
| 15 | erf | 2 | 15 | er | 6 |
| 16 | erfc | 2 | 16 | i | 8 |
| 17 | exp | 2 | 17 | sqrtpi | 3 |
| 18 | if | 3 | 18 | sum | 9 |
| 19 | main | 1 | 19 | Sum1 | 2 |
| 20 | printf | 2 | 20 | term | 5 |
| 21 | return | 2 | 21 | terms | 3 |
| 22 | scanf | 1 | 22 | tol | 2 |
| 23 |  |  | 23 | u | 4 |
| 24 |  |  | 24 | v | 4 |
| 25 |  |  | 25 | x | 17 |
| 26 |  |  | 26 | x2 | 7 |

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Оператор** | **Количество** | **№** | **Операнд** | **Количество** |
| 1 | push | 3 | 1 | -2079671268 | 2 |
| 2 | mov | 21 | 2 | 1073503224 | 2 |
| 3 | sub | 4 | 3 | -350469331 | 2 |
| 4 | movsd | 48 | 4 | 1058682594 | 2 |
| 5 | mulsd | 9 | 5 | 1072693248 | 1 |
| 6 | add | 1 | 6 | -2147483648 | 1 |
| 7 | addsd | 9 | 7 | 1073217536 | 1 |
| 8 | movapd | 10 | 8 | 12 | 3 |
| 9 | pxor | 8 | 9 | 0 | 12 |
| 10 | cvtsi2sd | 3 | 10 | "Arg? " | 1 |
| 11 | divsd | 5 | 11 | "%lf" | 1 |
| 12 | comisd | 3 | 12 | "X = %.8lf; Erf = %.12lf; Erfc = %.12lf\n" | 1 |
| 13 | ja | 1 | 13 | rbp | 6 |
| 14 | movq | 17 | 14 | rsp | 6 |
| 15 | xorpd | 2 | 15 | 64 | 2 |
| 16 | call | 7 | 16 | QWORD PTR [rbp-56] | 7 |
| 17 | leave | 3 | 17 | xmm0 | 105 |
| 18 | ret | 3 | 18 | QWORD PTR [rbp-32] | 12 |
| 19 | cmp | 2 | 19 | QWORD PTR [rbp-8] | 13 |
| 20 | jg | 1 | 20 | QWORD PTR [rbp-16] | 10 |
| 21 | lea | 1 | 21 | DWORD PTR [rbp-20] | 6 |
| 22 | jbe | 2 | 22 | 1 | 3 |
| 23 | jmp | 3 | 23 | QWORD PTR [rbp-40] | 5 |
| 24 | ucomisd | 2 | 24 | xmm1 | 42 |
| 25 | jp | 1 | 25 | xmm2 | 5 |
| 26 | jne | 2 | 26 | QWORD PTR .LC0[rip] | 8 |
| 27 | subsd | 2 | 27 | QWORD PTR .LC1[rip] | 1 |
| 28 |  |  | 28 | .L2 | 1 |
| 29 |  |  | 29 | QWORD PTR [rbp-64] | 2 |
| 30 |  |  | 30 | QWORD PTR .LC2[rip] | 2 |
| 31 |  |  | 31 | xmm4 | 3 |
| 32 |  |  | 32 | rax | 22 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Оператор** | **Число вхождений** | **№** | **Операнд** | **Число вхождений** |
|  |  |  | 33 | exp | 2 |
|  |  |  | 34 | QWORD PTR .LC3[rip] | 2 |
|  |  |  | 35 | QWORD PTR [rbp-24] | 3 |
|  |  |  | 36 | DWORD PTR [rbp-12] | 4 |
|  |  |  | 37 | .L5 | 1 |
|  |  |  | 38 | 32 | 1 |
|  |  |  | 39 | edi | 3 |
|  |  |  | 40 | OFFSET FLAT:.LC4 | 1 |
|  |  |  | 41 | printf | 2 |
|  |  |  | 42 | [rbp-32] | 7 |
|  |  |  | 43 | rsi | 1 |
|  |  |  | 44 | OFFSET FLAT:.LC5 | 1 |
|  |  |  | 45 | \_\_isoc99\_scanf | 1 |
|  |  |  | 46 | .L21 | 1 |
|  |  |  | 47 | .L10 | 1 |
|  |  |  | 48 | .L11 | 2 |
|  |  |  | 49 | .L13 | 2 |
|  |  |  | 50 | QWORD PTR .LC7[rip] | 1 |
|  |  |  | 51 | .L22 | 1 |
|  |  |  | 52 | erf | 1 |
|  |  |  | 53 | erfc | 1 |
|  |  |  | 54 | OFFSET FLAT:.LC8 | 1 |
|  |  |  | 55 | .L16 | 1 |

Продолжение таблицы 3

В таблице 4 представлены сводные результаты расчетных характеристик.

Таблица 4 – Результаты расчетных характеристик, посчитанные вручную

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Паскаль** | **Си** | **Ассемблер** |
| Число уникальных операторов (n1): | **20** | **22** | **27** |
| Число уникальных операндов (n2): | **34** | **26** | **55** |
| Общее число операторов  (N1): | *98* | *102* | *173* |

|  |  |  |  |
| --- | --- | --- | --- |
| Общее число операндов  (N2): | *121* | *106* | *331* |
| Алфавит (n): | 54 | 48 | 82 |
| Экспериментальная длина  программы (Nэ): | 219 | 208 | 504 |
| Теоретическая длина программы (Nт): | 259.412 | 220.319 | 446,35 |
| Объём программы (V): | 1260.32 | 1161.67 | 3204,2052 |
| Потенциальный объём  (V\*): | 19.6515 | 19.6515 | 19.6515 |
| Уровень программы (L): | 0.0155925 | 0.0169165 | 0.00613303 |
| Интеллект программы (I): | 35.414 | 25.9035 | 39,4383 |
| Работа по программированию (Е): | 80828.9 | 68670.8 | 528180,497 |
| Время кодирования (T): | 4490.49 | 3815.04 | 522818 |
| Уровень языка программирования (Lam): | 0.306415 | 0.332435 | 0.12052265 |
| Уровень ошибок (В): | 1 | 1 | 4 |

# Расчет метрик с помощью программы автоматизации

**Для программы на Pascal:**

Statistics for module omainPas.lxm

=====================================

The number of different operators : 20

The number of different operands : 34

The total number of operators : 98

The total number of operands : 121

Dictionary ( D) : 54

Length ( N) : 219

Length estimation ( ^N) : 259.412

Volume ( V) : 1260.32

Potential volume ( \*V) : 19.6515

Limit volume (\*\*V) : 38.2071

Programming level ( L) : 0.0155925

Programming level estimation ( ^L) : 0.0280992

Intellect ( I) : 35.414

Time of programming ( T) : 4490.49

Time estimation ( ^T) : 2951.63

Programming language level (lambda) : 0.306415

Work on programming ( E) : 80828.9

Error ( B) : 0.623146

Error estimation ( ^B) : 0.420107

Table:

====================================

Operators:

| 1 | 17 | ()

| 2 | 13 | \*

| 3 | 6 | +

| 4 | 4 | -

| 5 | 5 | /

| 6 | 3 | <

| 7 | 28 | =

| 8 | 1 | ClrScr

| 9 | 2 | const

| 10 | 2 | erf

| 11 | 2 | erfc

| 12 | 2 | exp

| 13 | 1 | for

| 14 | 3 | if

| 15 | 1 | program

| 16 | 1 | readln

| 17 | 2 | real

| 18 | 2 | repeat

| 19 | 1 | write

| 20 | 2 | writeln

Operands:

| 1 | 1 | ' Erf= '

| 2 | 1 | ', Erfc= '

| 3 | 1 | 'Arg? '

| 4 | 1 | 'X= '

| 5 | 1 | 0

| 6 | 3 | 0.0

| 7 | 2 | 1

| 8 | 8 | 1.0

| 9 | 1 | 1.0E-4

| 10 | 1 | 1.5

| 11 | 2 | 1.7724538

| 12 | 3 | 12

| 13 | 4 | 2.0

| 14 | 1 | 4

| 15 | 2 | 8

| 16 | 4 | done

| 17 | 6 | ec

| 18 | 6 | er

| 19 | 1 | erf

| 20 | 1 | erfc

| 21 | 1 | erfd3

| 22 | 1 | false

| 23 | 7 | i

| 24 | 4 | sqrtpi

| 25 | 10 | sum

| 26 | 3 | sum1

| 27 | 6 | term

| 28 | 3 | terms

| 29 | 2 | tol

| 30 | 1 | true

| 31 | 4 | u

| 32 | 4 | v

| 33 | 17 | x

| 34 | 8 | x2

Summary:

=====================================

The number of different operators : 20

The number of different operands : 34

The total number of operators : 98

The total number of operands : 121

Dictionary ( D) : 54

Length ( N) : 219

Length estimation ( ^N) : 259.412

Volume ( V) : 1260.32

Potential volume ( \*V) : 19.6515

Limit volume (\*\*V) : 38.2071

Programming level ( L) : 0.0155925

Programming level estimation ( ^L) : 0.0280992

Intellect ( I) : 35.414

Time of programming ( T) : 4490.49

Time estimation ( ^T) : 2951.63

Programming language level (lambda) : 0.306415

Work on programming ( E) : 80828.9

Error ( B) : 0.623146

Error estimation ( ^B) : 0.420107

**Для программы на языке C:**

# Statistics for module omainC.lxm

# =====================================

# The number of different operators : 22

# The number of different operands : 26

# The total number of operators : 102

# The total number of operands : 106

# Dictionary ( D) : 48

# Length ( N) : 208

# Length estimation ( ^N) : 220.319

# Volume ( V) : 1161.67

# Potential volume ( \*V) : 19.6515

# Limit volume (\*\*V) : 38.2071

# Programming level ( L) : 0.0169165

# Programming level estimation ( ^L) : 0.0222985

# Intellect ( I) : 25.9035

# Time of programming ( T) : 3815.04

# Time estimation ( ^T) : 3065.67

# Programming language level (lambda) : 0.332435

# Work on programming ( E) : 68670.8

# Error ( B) : 0.558976

# Error estimation ( ^B) : 0.387224

# Table:

# ====================================

# Operators:

# | 1 | 15 | ()

# | 2 | 13 | \*

# | 3 | 6 | +

# | 4 | 9 | ,

# | 5 | 2 | -

# | 6 | 1 | --

# | 7 | 5 | /

# | 8 | 3 | <

# | 9 | 25 | =

# | 10 | 1 | ==

# | 11 | 1 | >=

# | 12 | 1 | \_&

# | 13 | 2 | \_-

# | 14 | 3 | dowhile

# | 15 | 2 | erf

# | 16 | 2 | erfc

# | 17 | 2 | exp

# | 18 | 3 | if

# | 19 | 1 | main

# | 20 | 2 | printf

# | 21 | 2 | return

# | 22 | 1 | scanf

# Operands:

# | 1 | 1 | "%lf"

# | 2 | 1 | "Arg? "

# | 3 | 1 | "X = %.8lf; Erf = %.12lf; Erfc = %.12lf\n"

# | 4 | 2 | 0

# | 5 | 3 | 0.0

# | 6 | 3 | 1

# | 7 | 8 | 1.0

# | 8 | 1 | 1.0E-4

# | 9 | 1 | 1.5

# | 10 | 1 | 1.7724538

# | 11 | 1 | 12

# | 12 | 4 | 2.0

# | 13 | 3 | done

# | 14 | 6 | ec

# | 15 | 6 | er

# | 16 | 8 | i

# | 17 | 3 | sqrtpi

# | 18 | 9 | sum

# | 19 | 2 | sum1

# | 20 | 5 | term

# | 21 | 3 | terms

# | 22 | 2 | tol

# | 23 | 4 | u

# | 24 | 4 | v

# | 25 | 17 | x

# | 26 | 7 | x2

# Summary:

# =====================================

# The number of different operators : 22

# The number of different operands : 26

# The total number of operators : 102

# The total number of operands : 106

# Dictionary ( D) : 48

# Length ( N) : 208

# Length estimation ( ^N) : 220.319

# Volume ( V) : 1161.67

# Potential volume ( \*V) : 19.6515

# Limit volume (\*\*V) : 38.2071

# Programming level ( L) : 0.0169165

# Programming level estimation ( ^L) : 0.0222985

# Intellect ( I) : 25.9035

# Time of programming ( T) : 3815.04

# Time estimation ( ^T) : 3065.67

# Programming language level (lambda) : 0.332435

# Work on programming ( E) : 68670.8

# Error ( B) : 0.558976

# Error estimation ( ^B) : 0.387224

# Вывод

Метрические характеристики программ, написанных на языках C и Паскаль, выглядят похожим образом, так как C и Паскаль являются языками примерно одинаково высокого уровня. В следствии того, что язык ассемблера является языком программирования более низкого уровня, характеристики программы, написанной на нем, значительно отличаются в большую сторону. Характеристики были посчитаны вручную и автоматически.

**ПРИЛОЖЕНИЕ А.**

**КОД ПРОГРАММЫ НА ЯЗЫКЕ ПАСКАЛЬ**

program erfd3;

{ 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;

tol = 1.0E-4;

var x2,sum,sum1,term: real;

i : integer;

begin

x2:=x\*x;

sum:=x;

term:=x;

i:=0;

repeat

i:=i+1;

sum1:=sum;

term:=2.0\*term\*x2/(1.0+2.0\*i);

sum:=term+sum1

until term<tol\*sum;

erf:=2.0\*sum\*exp(-x2)/sqrtpi

end; { erf }

function erfc(x: real): real;

{ complement of error function }

const sqrtpi = 1.7724538;

terms = 12;

var x2,u,v,sum : real;

i : integer;

begin

x2:=x\*x;

v:=1.0/(2.0\*x2);

u:=1.0+v\*(terms+1.0);

for i:=terms downto 1 do

begin

sum:=1.0+i\*v/u;

u:=sum

end;

erfc:=exp(-x2)/(x\*sum\*sqrtpi)

end; { ercf }

begin { main }

ClrScr;

done:=false;

writeln;

repeat

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:12:8,', Erfc= ',ec:12)

end { if }

until done

end.

**ПРИЛОЖЕНИЕ Б.**

**ПРОГРАММА НА ЯЗЫКЕ СИ**

#include <math.h>

#include <stdio.h>

const double sqrtpi = 1.7724538;

const double tol = 1.0E-4;

const int terms = 12;

// infinite series expansion of the Gaussian error function

double erf (double x) {

double x2 = x \* x;

double sum = x;

double term = x;

int i = 0;

do {

i = i + 1;

double sum1 = sum;

term = 2.0 \* term \* x2 / (1.0 + 2.0 \* i);

sum = term + sum1;

} while (term < tol \* sum);

return 2.0 \* sum \* exp(-x2) / sqrtpi;

}

// complement of error function

double erfc (double x) {

double x2,u,v,sum;

x2 = x \* x;

v = 1.0 / (2.0 \* x2);

u = 1.0 + v \* (terms + 1.0);

int i = terms;

do {

sum = 1.0 + i \* v / u;

u = sum;

i--;

} while (i >= 1);

return exp(-x2) / (x \* sum \* sqrtpi);

}

// evaluation of the gaussian error function

int main () {

double x, er, ec;

int done = 1;

do {

printf("Arg? ");

scanf("%lf", &x);

if (x < 0.0) done = 0;

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;

}

}

printf("X = %.8lf; Erf = %.12lf; Erfc = %.12lf\n", x, er, ec);

}

} while (done);

}

**ПРИЛОЖЕНИЕ В.**

**ПРОГРАММА НА ЯЗЫКЕ АССЕМБЛЕР**

. # Generated with this tool: https://gcc.godbolt.org

sqrtpi:

.long -2079671268

.long 1073503224

tol:

.long -350469331

.long 1058682594

terms:

.long 12

erf:

push rbp

mov rbp, rsp

sub rsp, 64

movsd QWORD PTR [rbp-56], xmm0

movsd xmm0, QWORD PTR [rbp-56]

mulsd xmm0, xmm0

movsd QWORD PTR [rbp-32], xmm0

movsd xmm0, QWORD PTR [rbp-56]

movsd QWORD PTR [rbp-8], xmm0

movsd xmm0, QWORD PTR [rbp-56]

movsd QWORD PTR [rbp-16], xmm0

mov DWORD PTR [rbp-20], 0

.L2:

add DWORD PTR [rbp-20], 1

movsd xmm0, QWORD PTR [rbp-8]

movsd QWORD PTR [rbp-40], xmm0

movsd xmm0, QWORD PTR [rbp-16]

addsd xmm0, xmm0

movapd xmm1, xmm0

mulsd xmm1, QWORD PTR [rbp-32]

pxor xmm0, xmm0

cvtsi2sd xmm0, DWORD PTR [rbp-20]

movapd xmm2, xmm0

addsd xmm2, xmm0

movsd xmm0, QWORD PTR .LC0[rip]

addsd xmm2, xmm0

divsd xmm1, xmm2

movapd xmm0, xmm1

movsd QWORD PTR [rbp-16], xmm0

movsd xmm0, QWORD PTR [rbp-16]

addsd xmm0, QWORD PTR [rbp-40]

movsd QWORD PTR [rbp-8], xmm0

movsd xmm0, QWORD PTR .LC1[rip]

mulsd xmm0, QWORD PTR [rbp-8]

comisd xmm0, QWORD PTR [rbp-16]

ja .L2

movsd xmm0, QWORD PTR [rbp-8]

addsd xmm0, xmm0

movsd QWORD PTR [rbp-64], xmm0

movsd xmm0, QWORD PTR [rbp-32]

movq xmm1, QWORD PTR .LC2[rip]

movapd xmm4, xmm0

xorpd xmm4, xmm1

movq rax, xmm4

movq xmm0, rax

call exp

mulsd xmm0, QWORD PTR [rbp-64]

movsd xmm1, QWORD PTR .LC3[rip]

divsd xmm0, xmm1

movq rax, xmm0

movq xmm0, rax

leave

ret

erfc:

push rbp

mov rbp, rsp

sub rsp, 64

movsd QWORD PTR [rbp-56], xmm0

movsd xmm0, QWORD PTR [rbp-56]

mulsd xmm0, xmm0

movsd QWORD PTR [rbp-24], xmm0

movsd xmm0, QWORD PTR [rbp-24]

movapd xmm1, xmm0

addsd xmm1, xmm0

movsd xmm0, QWORD PTR .LC0[rip]

divsd xmm0, xmm1

movsd QWORD PTR [rbp-32], xmm0

mov eax, 12

pxor xmm1, xmm1

cvtsi2sd xmm1, eax

movsd xmm0, QWORD PTR .LC0[rip]

addsd xmm0, xmm1

movapd xmm1, xmm0

mulsd xmm1, QWORD PTR [rbp-32]

movsd xmm0, QWORD PTR .LC0[rip]

addsd xmm0, xmm1

movsd QWORD PTR [rbp-8], xmm0

mov DWORD PTR [rbp-12], 12

.L5:

pxor xmm0, xmm0

cvtsi2sd xmm0, DWORD PTR [rbp-12]

mulsd xmm0, QWORD PTR [rbp-32]

movapd xmm1, xmm0

divsd xmm1, QWORD PTR [rbp-8]

movsd xmm0, QWORD PTR .LC0[rip]

addsd xmm0, xmm1

movsd QWORD PTR [rbp-40], xmm0

movsd xmm0, QWORD PTR [rbp-40]

movsd QWORD PTR [rbp-8], xmm0

sub DWORD PTR [rbp-12], 1

cmp DWORD PTR [rbp-12], 0

jg .L5

movsd xmm0, QWORD PTR [rbp-24]

movq xmm1, QWORD PTR .LC2[rip]

xorpd xmm0, xmm1

movq rax, xmm0

movq xmm0, rax

call exp

movq rax, xmm0

movsd xmm0, QWORD PTR [rbp-56]

movapd xmm1, xmm0

mulsd xmm1, QWORD PTR [rbp-40]

movsd xmm0, QWORD PTR .LC3[rip]

mulsd xmm0, xmm1

movq xmm1, rax

divsd xmm1, xmm0

movq rax, xmm1

movq xmm0, rax

leave

ret

.LC4:

.string "Arg? "

.LC5:

.string "%lf"

.LC8:

.string "X = %.8lf; Erf = %.12lf; Erfc = %.12lf\n"

main:

push rbp

mov rbp, rsp

sub rsp, 32

mov DWORD PTR [rbp-20], 1

.L16:

mov edi, OFFSET FLAT:.LC4

mov eax, 0

call printf

lea rax, [rbp-32]

mov rsi, rax

mov edi, OFFSET FLAT:.LC5

mov eax, 0

call \_\_isoc99\_scanf

movsd xmm1, QWORD PTR [rbp-32]

pxor xmm0, xmm0

comisd xmm0, xmm1

jbe .L21

mov DWORD PTR [rbp-20], 0

jmp .L10

.L21:

movsd xmm0, QWORD PTR [rbp-32]

pxor xmm1, xmm1

ucomisd xmm0, xmm1

jp .L11

pxor xmm1, xmm1

ucomisd xmm0, xmm1

jne .L11

pxor xmm0, xmm0

movsd QWORD PTR [rbp-8], xmm0

movsd xmm0, QWORD PTR .LC0[rip]

movsd QWORD PTR [rbp-16], xmm0

jmp .L13

.L11:

movsd xmm1, QWORD PTR [rbp-32]

movsd xmm0, QWORD PTR .LC7[rip]

comisd xmm0, xmm1

jbe .L22

mov rax, QWORD PTR [rbp-32]

movq xmm0, rax

call erf

movq rax, xmm0

mov QWORD PTR [rbp-8], rax

movsd xmm0, QWORD PTR .LC0[rip]

subsd xmm0, QWORD PTR [rbp-8]

movsd QWORD PTR [rbp-16], xmm0

jmp .L13

.L22:

mov rax, QWORD PTR [rbp-32]

movq xmm0, rax

call erfc

movq rax, xmm0

mov QWORD PTR [rbp-16], rax

movsd xmm0, QWORD PTR .LC0[rip]

subsd xmm0, QWORD PTR [rbp-16]

movsd QWORD PTR [rbp-8], xmm0

.L13:

mov rax, QWORD PTR [rbp-32]

movsd xmm1, QWORD PTR [rbp-16]

movsd xmm0, QWORD PTR [rbp-8]

movapd xmm2, xmm1

movapd xmm1, xmm0

movq xmm0, rax

mov edi, OFFSET FLAT:.LC8

mov eax, 3

call printf

.L10:

cmp DWORD PTR [rbp-20], 0

jne .L16

mov eax, 0

leave

ret

.LC0:

.long 0

.long 1072693248

.LC1:

.long -350469331

.long 1058682594

.LC2:

.long 0

.long -2147483648

.long 0

.long 0

.LC3:

.long -2079671268

.long 1073503224

.LC7:

.long 0

.long 1073217536