**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. 6304 |  | Блинникова Ю.И. |
| Преподаватель |  | Кирьянчиков В.А. |

Санкт-Петербург

2020

# 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

Программа на языке Паскаль, С и Assembler представлены в приложениях A, 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** | **Число вхождений** | **№** | **Операнд** | **Число вхождений** |
| 1 | ; | 32 | 1 | 1.0E-6 | 1 |
| 2 | \* | 11 | 2 | 2 | 6 |
| 3 | + | 9 | 3 | tol | 4 |
| 4 | - | 8 | 4 | sum | 10 |
| 5 | / | 8 | 5 | upper | 8 |
| 6 | () | 24 | 6 | lower | 10 |
| 7 | <= | 1 | 7 | x | 7 |
| 8 | <> | 1 | 8 | i | 2 |
| 9 | = | 20 | 9 | delta\_x | 8 |
| 10 | begin…end | 5 | 10 | even\_sum | 5 |
| 11 | repeat…until | 1 | 11 | odd\_sum | 8 |
| 12 | abs | 2 | 12 | end\_sum | 4 |
| 13 | and | 1 | 13 | end\_cor | 3 |
| 14 | simps | 2 | 14 | sum1 | 4 |
| 15 | fx | 6 | 15 | pieces | 7 |
| 16 | dfx | 4 | 16 | 0.0 | 2 |
| 17 | exp | 2 | 17 | 4.0 | 1 |
| 18 | div | 1 | 18 | simp1 | 1 |
| 19 | for…to…do | 1 | 19 | 1.0 | 2 |
| 20 | writeln | 2 | 20 | 7.0 | 1 |
| 21 | chr | 1 | 21 | 14.0 | 1 |
|  |  |  | 22 | 16.0 | 1 |
|  |  |  | 23 | 15.0 | 1 |
|  |  |  | 24 | 9.0 | 1 |
|  |  |  | 25 | 'area= ' | 1 |
|  |  |  | 26 | 1 | 1 |
|  |  |  | 27 | 2.0 | 1 |
|  |  |  | 28 | 3.0 | 1 |

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Оператор** | **Число вхождений** | **№** | **Операнд** | **Число вхождений** |
| 1 | ; | 33 | 1 | 1.0E-6 | 1 |
| 2 | \* | 14 | 2 | 2 | 3 |
| 3 | + | 9 | 3 | tol | 4 |
| 4 | - | 5 | 4 | sum | 10 |
| 5 | / | 8 | 5 | upper | 8 |
| 6 | () | 14 | 6 | lower | 9 |
| 7 | <= | 2 | 7 | x | 7 |
| 8 | <> | 1 | 8 | i | 4 |
| 9 | = | 22 | 9 | delta\_x | 8 |
| 10 | main | 1 | 10 | even\_sum | 5 |
| 11 | do…while | 1 | 11 | odd\_sum | 8 |
| 12 | abs | 2 | 12 | end\_sum | 4 |
| 13 | and | 1 | 13 | end\_cor | 3 |
| 14 | simps | 2 | 14 | sum1 | 4 |
| 15 | fx | 6 | 15 | pieces | 7 |
| 16 | dfx | 4 | 16 | 0.0 | 2 |
| 17 | exp | 2 | 17 | 4.0 | 1 |
| 18 | & | 1 | 18 | simp1 | 1 |
| 19 | ++ | 1 | 19 | 1.0 | 5 |
| 20 | printf | 2 | 20 | 7.0 | 1 |
| 21 | return | 4 | 21 | 14.0 | 1 |
| 22 | != | 1 | 22 | 16.0 | 1 |
| 23 | for | 1 | 23 | 15.0 | 1 |
|  |  |  | 24 | 9.0 | 1 |
|  |  |  | 25 | 'area= ' | 1 |
|  |  |  | 26 | 1 | 1 |
|  |  |  | 27 | 2.0 | 4 |
|  |  |  | 28 | 3.0 | 1 |
|  |  |  | 29 | res | 3 |

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Оператор** | **Число вхождений** | **№** | **Операнд** | **Число вхождений** |
| 1 | pushq | 4 | 1 | rbp | 102 |
| 2 | movq | 27 | 2 | rsp | 8 |
| 3 | subq | 4 | 3 | 16 | 19 |
| 4 | movsd | 67 | 4 | xmm0 | 106 |
| 5 | leaq | 1 | 5 | -8(%rbp) | 9 |
| 6 | xorpd | 3 | 6 | xmm1 | 37 |
| 7 | divsd | 7 | 7 | rax | 20 |
| 8 | call | 10 | 8 | -16(%rbp) | 8 |
| 9 | leave | 4 | 9 | -8(%rbp) | 9 |
| 10 | ret | 4 | 10 | 112 | 1 |
| 11 | movapd | 6 | 11 | -72(%rbp) | 7 |
| 12 | movl | 11 | 12 | -80(%rbp) | 5 |
| 13 | subsd | 5 | 13 | -88(%rbp) | 2 |
| 14 | cvtsi2sd | 3 | 14 | xmm3 | 2 |
| 15 | addsd | 10 | 15 | xmm2 | 8 |
| 16 | pxor | 4 | 16 | -96(%rbp) | 9 |
| 17 | movapd | 6 | 17 | 2 | 1 |
| 18 | mulsd | 9 | 18 | -64(%rbp) | 5 |
| 19 | sall | 1 | 19 | -40(%rbp) | 11 |
| 20 | addl | 2 | 20 | fx | 16 |
| 21 | shrl | 1 | 21 | exp@PLT | 2 |
| 22 | cmpl | 2 | 22 | rax | 20 |
| 23 | sarl | 3 | 23 | -56(%rbp) | 4 |
| 24 | setp | 1 | 24 | -104(%rbp) | 10 |
| 25 | ucomisd | 2 | 25 | -112(%rbp) | 4 |
| 26 | subl | 2 | 26 | -32(%rbp) | 5 |
| 27 | cmove | 1 | 27 | dfx | 7 |
| 28 | cvttsd2si | 2 | 28 | xmm4 | 3 |
| 29 | xorl | 2 | 29 | -24(%rbp) | 4 |
| 30 | setle | 1 | 30 | -48(%rbp) | 7 |
| 31 | testb | 1 | 31 | 1 | 4 |
| 32 | jne | 1 | 32 | -60(%rbp) | 4 |
| 33 |  |  | 33 | eax | 18 |
| 34 |  |  | 34 | edx | 8 |
| 35 |  |  | 35 | 31 | 3 |
| 36 |  |  | 36 | -96(%rbp) | 9 |
| 37 |  |  | 37 | al | 4 |
| 38 |  |  | 38 | esi | 3 |
| 39 |  |  | 39 | 48 | 1 |
| 40 |  |  | 40 | rdi | 1 |
| 41 |  |  | 41 | printf@PLT | 1 |

В таблице 4 представлены сводные результаты расчетных характеристик вручную.

Таблица 4 – Результаты расчетных характеристик вручную

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Паскаль** | **Си** | **Ассемблер** |
| Число уникальных операторов (n1): | 21 | 23 | 32 |
| Число уникальных операндов (n2): | 28 | 29 | 41 |
| Общее число операторов (N1): | 141 | 137 | 207 |
| Общее число операндов (N2): | 102 | 109 | 507 |
| Алфавит (n): | 49 | 52 | 73 |
| Экспериментальная длина программы (Nэ): | 243 | 246 | 714 |
| Теоретическая длина программы (Nт): | 226.844603 | 244.923373 | 379.659632 |
| Объём программы (V): | 1364.374492 | 1402.308170 | 4419.534735 |
| Потенциальный объём (V\*): | 11.609640 | 11.609640 | 11.609640 |
| Уровень программы (L): | 0.008509 | 0.008278 | 0.002626 |
| Ожидание уровня программы (L^): | 0.026143 | 0.023135 | 0.005054 |
| Интеллект программы (I): | 35.669921 | 32.442709 | 22.337392 |
| Работа по программированию (Е): | 160342.411881 | 169382.351661 | 168241.965091 |
| Ожидание времени кодирования (T^): | 16034.241188 | 16938.235166 | 87442.1116357 |
| Время кодирования (T): | 5218.732432 | 6061.356179 | 0.030497 |
| Уровень языка программирования (Lam): | 0.098787 | 0.096115 | 11.048836 |
| Уровень ошибок (В): | 3.410936 | 3.505770 |  |

# Расчет метрик с помощью программы автоматизации

Для программы на Pascal:

Statistics for module output\_pascal.lxm

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

The number of different operators : 25

The number of different operands : 31

The total number of operators : 164

The total number of operands : 105

Dictionary ( D) : 56

Length ( N) : 269

Length estimation ( ^N) : 269.676

Volume ( V) : 1562.18

Potential volume ( \*V) : 19.6515

Limit volume (\*\*V) : 38.2071

Programming level ( L) : 0.0125795

Programming level estimation ( ^L) : 0.023619

Intellect ( I) : 36.8972

Time of programming ( T) : 6899.12

Time estimation ( ^T) : 3683.72

Programming language level (lambda) : 0.247207

Work on programming ( E) : 124184

Error ( B) : 0.829703

Error estimation ( ^B) : 0.520726

Table:

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

Operators:

| 1 | 24 | ()

| 2 | 11 | \*

| 3 | 9 | +

| 4 | 8 | -

| 5 | 8 | /

| 6 | 47 | ;

| 7 | 1 | <=

| 8 | 1 | <>

| 9 | 20 | =

| 10 | 2 | abs

| 11 | 1 | and

| 12 | 1 | chr

| 13 | 1 | const

| 14 | 3 | dfx

| 15 | 2 | exp

| 16 | 1 | for

| 17 | 2 | function

| 18 | 5 | fx

| 19 | 2 | integer

| 20 | 1 | procedure

| 21 | 1 | program

| 22 | 8 | real

| 23 | 1 | repeat

| 24 | 2 | simps

| 25 | 2 | writeln

Operands:

| 1 | 1 | 'area= '

| 2 | 2 | 0.0

| 3 | 1 | 1

| 4 | 2 | 1.0

| 5 | 1 | 1.0E-6

| 6 | 1 | 14.0

| 7 | 1 | 15.0

| 8 | 1 | 16.00

| 9 | 6 | 2

| 10 | 1 | 2.0

| 11 | 1 | 3.0

| 12 | 1 | 4.0

| 13 | 1 | 7

| 14 | 1 | 7.0

| 15 | 1 | 9.0

| 16 | 8 | delta\_x

| 17 | 1 | dfx

| 18 | 3 | end\_cor

| 19 | 4 | end\_sum

| 20 | 5 | even\_sum

| 21 | 1 | fx

| 22 | 2 | i

| 23 | 10 | lower

| 24 | 8 | odd\_sum

| 25 | 7 | pieces

| 26 | 1 | simp1

| 27 | 10 | sum

| 28 | 4 | sum1

| 29 | 4 | tol

| 30 | 8 | upper

| 31 | 7 | x

Для программы на Cи:

Statistics for module output\_lab1\_c.lxm

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

The number of different operators : 27

The number of different operands : 29

The total number of operators : 168

The total number of operands : 110

Dictionary ( D) : 56

Length ( N) : 278

Length estimation ( ^N) : 269.263

Volume ( V) : 1614.44

Potential volume ( \*V) : 19.6515

Limit volume (\*\*V) : 38.2071

Programming level ( L) : 0.0121723

Programming level estimation ( ^L) : 0.0195286

Intellect ( I) : 31.5279

Time of programming ( T) : 7368.49

Time estimation ( ^T) : 4448.48

Programming language level (lambda) : 0.239204

Work on programming ( E) : 132633

Error ( B) : 0.866921

Error estimation ( ^B) : 0.538148

Table:

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

Operators:

| 1 | 1 | !=

| 2 | 1 | &

| 3 | 14 | ()

| 4 | 14 | \*

| 5 | 9 | +

| 6 | 1 | ++

| 7 | 13 | ,

| 8 | 5 | -

| 9 | 8 | /

| 10 | 33 | ;

| 11 | 2 | <=

| 12 | 22 | =

| 13 | 3 | \_-

| 14 | 2 | abs

| 15 | 1 | const

| 16 | 3 | dfx

| 17 | 15 | double

| 18 | 1 | dowhile

| 19 | 2 | exp

| 20 | 1 | for

| 21 | 5 | fx

| 22 | 3 | int

| 23 | 1 | main

| 24 | 1 | printf

| 25 | 4 | return

| 26 | 2 | simps

| 27 | 1 | void

Operands:

| 1 | 1 | "area= %lf "

| 2 | 1 | 0

| 3 | 4 | 0.0

| 4 | 1 | 1

| 5 | 5 | 1.0

| 6 | 1 | 1.0E-6

| 7 | 1 | 14.0

| 8 | 1 | 15.0

| 9 | 1 | 16.00

| 10 | 3 | 2

| 11 | 4 | 2.0

| 12 | 1 | 3.0

| 13 | 1 | 4.0

| 14 | 1 | 7.0

| 15 | 1 | 9.0

| 16 | 8 | delta\_x

| 17 | 3 | end\_cor

| 18 | 4 | end\_sum

| 19 | 5 | even\_sum

| 20 | 4 | i

| 21 | 9 | lower

| 22 | 8 | odd\_sum

| 23 | 7 | pieces

| 24 | 3 | res

| 25 | 10 | sum

| 26 | 4 | sum1

| 27 | 4 | tol

| 28 | 7 | upper

| 29 | 7 | x

# 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. 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 simp1;

{ integration by Simpson's method }

const tol = 1.0E-6;

var sum,upper,lower : real;

function fx(x: real): real;

begin

fx:=exp(-x/2)

end; { function fx }

function dfx(x: real): real;

begin

dfx:=-(exp(-x/2))/2

end; { function fx }

procedure simps(

lower,upper,tol : real;

var sum : real);

{ numerical integration by Simpson's rule }

{ function is fx, limits are lower and upper }

{ with number of regions equal to pieces }

{ partition is delta\_x, answer is sum }

var i : integer;

x,delta\_x,even\_sum,

odd\_sum,end\_sum,

end\_cor,sum1 : real;

pieces : integer;

begin

pieces:=2;

delta\_x:=(upper-lower)/pieces;

odd\_sum:=fx(lower+delta\_x);

even\_sum:=0.0;

end\_sum:=fx(lower)+fx(upper);

end\_cor:=dfx(lower)-dfx(upper);

sum:=(end\_sum+4.0\*odd\_sum)\*delta\_x/3.0;

repeat

pieces:=pieces\*2;

sum1:=sum;

delta\_x:=(upper-lower)/pieces;

even\_sum:=even\_sum+odd\_sum;

odd\_sum:=0.0;

for i:=1 to pieces div 2 do

begin

x:=lower+delta\_x\*(2.0\*i-1.0);

odd\_sum:=odd\_sum+fx(x)

end;

sum:=(7.0\*end\_sum+14.0\*even\_sum+16.00\*odd\_sum

+end\_cor\*delta\_x)\*delta\_x/15.0;

until (sum<>sum1) and (abs(sum-sum1)<=abs(tol\*sum))

end; { simps }

begin { main program }

lower:=1.0;

upper:=9.0;

simps(lower,upper,tol,sum);

writeln;

writeln(chr(7),'area= ',sum)

end.

**ПРИЛОЖЕНИЕ Б.**

**ПРОГРАММА НА ЯЗЫКЕ СИ**

#include <stdio.h>

#include <stdlib.h>

#include "math.h"

const double tol= 1.0E-6;

double fx(double x){

return exp(-1.0\*x/2.0);

}

double dfx(double x){

return (-1.0\*exp(-1.0\*x/2.0)/2.0);

}

double simps(double lower,double upper,double tol,double sum){

double x,delta\_x,even\_sum,odd\_sum,end\_sum,end\_cor,sum1;

int pieces;

pieces=2;

delta\_x=(upper-lower)/pieces;

odd\_sum=fx(lower+delta\_x);

even\_sum=0.0;

end\_sum=fx(lower)+fx(upper);

end\_cor=dfx(lower)-dfx(upper);

sum=(end\_sum+4.0\*odd\_sum)\*delta\_x/3.0;

do{

pieces=pieces\*2;

sum1=sum;

delta\_x=(upper-lower)/pieces;

even\_sum=even\_sum+odd\_sum;

odd\_sum=0.0;

for (int i=1;i<=pieces/2;i++){

x=lower+delta\_x\*(2.0\*i-1.0);

odd\_sum=odd\_sum+fx(x);

}

sum=(7.0\*end\_sum+14.0\*even\_sum+16.00\*odd\_sum

+end\_cor\*delta\_x)\*delta\_x/15.0;

}

while ((sum!=sum1) & (abs(sum-sum1)<=abs(tol\*sum)));

return sum;

}

int main(void) {

double sum=0.0;

double lower=1.0;

double upper=9.0;

double res =0.0;

res=simps(lower,upper,tol,sum);

printf("area= %lf ",res);

return 0;

}

# ПРИЛОЖЕНИЕ В.

# ПРОГРАММА НА ЯЗЫКЕ АССЕМБЛЕР

.file "lab1.c"

.text

.globl tol

.section .rodata

.align 8

.type tol, @object

.size tol, 8

tol:

.long 2696277389

.long 1051772663

.text

.globl fx

.type fx, @function

fx:

.LFB5:

.cfi\_startproc

pushq %rbp

.cfi\_def\_cfa\_offset 16

.cfi\_offset 6, -16

movq %rsp, %rbp

.cfi\_def\_cfa\_register 6

subq $16, %rsp

movsd %xmm0, -8(%rbp)

movsd -8(%rbp), %xmm1

movq .LC0(%rip), %xmm0

xorpd %xmm1, %xmm0

movsd .LC1(%rip), %xmm1

divsd %xmm1, %xmm0

call exp@PLT

movq %xmm0, %rax

movq %rax, -16(%rbp)

movsd -16(%rbp), %xmm0

leave

.cfi\_def\_cfa 7, 8

ret

.cfi\_endproc

.LFE5:

.size fx, .-fx

.globl dfx

.type dfx, @function

dfx:

.LFB6:

.cfi\_startproc

pushq %rbp

.cfi\_def\_cfa\_offset 16

.cfi\_offset 6, -16

movq %rsp, %rbp

.cfi\_def\_cfa\_register 6

subq $16, %rsp

movsd %xmm0, -8(%rbp)

movsd -8(%rbp), %xmm1

movq .LC0(%rip), %xmm0

xorpd %xmm1, %xmm0

movsd .LC1(%rip), %xmm1

divsd %xmm1, %xmm0

call exp@PLT

movapd %xmm0, %xmm1

movq .LC0(%rip), %xmm0

xorpd %xmm1, %xmm0

movsd .LC1(%rip), %xmm1

divsd %xmm1, %xmm0

leave

.cfi\_def\_cfa 7, 8

ret

.cfi\_endproc

.LFE6:

.size dfx, .-dfx

.globl simps

.type simps, @function

simps:

.LFB7:

.cfi\_startproc

pushq %rbp

.cfi\_def\_cfa\_offset 16

.cfi\_offset 6, -16

movq %rsp, %rbp

.cfi\_def\_cfa\_register 6

subq $112, %rsp

movsd %xmm0, -72(%rbp)

movsd %xmm1, -80(%rbp)

movsd %xmm2, -88(%rbp)

movsd %xmm3, -96(%rbp)

movl $2, -64(%rbp)

movsd -80(%rbp), %xmm0

subsd -72(%rbp), %xmm0

cvtsi2sd -64(%rbp), %xmm1

divsd %xmm1, %xmm0

movsd %xmm0, -40(%rbp)

movsd -72(%rbp), %xmm0

addsd -40(%rbp), %xmm0

call fx

movq %xmm0, %rax

movq %rax, -48(%rbp)

pxor %xmm0, %xmm0

movsd %xmm0, -56(%rbp)

movq -72(%rbp), %rax

movq %rax, -104(%rbp)

movsd -104(%rbp), %xmm0

call fx

movsd %xmm0, -104(%rbp)

movq -80(%rbp), %rax

movq %rax, -112(%rbp)

movsd -112(%rbp), %xmm0

call fx

addsd -104(%rbp), %xmm0

movsd %xmm0, -32(%rbp)

movq -72(%rbp), %rax

movq %rax, -104(%rbp)

movsd -104(%rbp), %xmm0

call dfx

movsd %xmm0, -104(%rbp)

movq -80(%rbp), %rax

movq %rax, -112(%rbp)

movsd -112(%rbp), %xmm0

call dfx

movsd -104(%rbp), %xmm4

subsd %xmm0, %xmm4

movapd %xmm4, %xmm0

movsd %xmm0, -24(%rbp)

movsd -48(%rbp), %xmm1

movsd .LC3(%rip), %xmm0

mulsd %xmm1, %xmm0

addsd -32(%rbp), %xmm0

mulsd -40(%rbp), %xmm0

movsd .LC4(%rip), %xmm1

divsd %xmm1, %xmm0

movsd %xmm0, -96(%rbp)

.L8:

sall -64(%rbp)

movsd -96(%rbp), %xmm0

movsd %xmm0, -16(%rbp)

movsd -80(%rbp), %xmm0

subsd -72(%rbp), %xmm0

cvtsi2sd -64(%rbp), %xmm1

divsd %xmm1, %xmm0

movsd %xmm0, -40(%rbp)

movsd -56(%rbp), %xmm0

addsd -48(%rbp), %xmm0

movsd %xmm0, -56(%rbp)

pxor %xmm0, %xmm0

movsd %xmm0, -48(%rbp)

movl $1, -60(%rbp)

jmp .L6

.L7:

cvtsi2sd -60(%rbp), %xmm0

addsd %xmm0, %xmm0

movsd .LC5(%rip), %xmm1

subsd %xmm1, %xmm0

mulsd -40(%rbp), %xmm0

movsd -72(%rbp), %xmm1

addsd %xmm1, %xmm0

movsd %xmm0, -8(%rbp)

movq -8(%rbp), %rax

movq %rax, -104(%rbp)

movsd -104(%rbp), %xmm0

call fx

movapd %xmm0, %xmm1

movsd -48(%rbp), %xmm0

addsd %xmm1, %xmm0

movsd %xmm0, -48(%rbp)

addl $1, -60(%rbp)

.L6:

movl -64(%rbp), %eax

movl %eax, %edx

shrl $31, %edx

addl %edx, %eax

sarl %eax

cmpl %eax, -60(%rbp)

jle .L7

movsd -32(%rbp), %xmm1

movsd .LC6(%rip), %xmm0

mulsd %xmm0, %xmm1

movsd -56(%rbp), %xmm2

movsd .LC7(%rip), %xmm0

mulsd %xmm2, %xmm0

addsd %xmm0, %xmm1

movsd -48(%rbp), %xmm2

movsd .LC8(%rip), %xmm0

mulsd %xmm2, %xmm0

addsd %xmm0, %xmm1

movsd -24(%rbp), %xmm0

mulsd -40(%rbp), %xmm0

addsd %xmm1, %xmm0

mulsd -40(%rbp), %xmm0

movsd .LC9(%rip), %xmm1

divsd %xmm1, %xmm0

movsd %xmm0, -96(%rbp)

movsd -96(%rbp), %xmm0

ucomisd -16(%rbp), %xmm0

setp %al

movl $1, %edx

movsd -96(%rbp), %xmm0

ucomisd -16(%rbp), %xmm0

movl %edx, %esi

cmove %eax, %esi

movsd -96(%rbp), %xmm0

subsd -16(%rbp), %xmm0

cvttsd2si %xmm0, %eax

movl %eax, %ecx

sarl $31, %ecx

xorl %ecx, %eax

movl %eax, %edx

subl %ecx, %edx

movsd -88(%rbp), %xmm0

mulsd -96(%rbp), %xmm0

cvttsd2si %xmm0, %eax

movl %eax, %ecx

sarl $31, %ecx

xorl %ecx, %eax

subl %ecx, %eax

cmpl %eax, %edx

setle %al

andl %esi, %eax

testb %al, %al

jne .L8

movsd -96(%rbp), %xmm0

leave

.cfi\_def\_cfa 7, 8

ret

.cfi\_endproc

.LFE7:

.size simps, .-simps

.section .rodata

.LC12:

.string "area= %lf "

.text

.globl main

.type main, @function

main:

.LFB8:

.cfi\_startproc

pushq %rbp

.cfi\_def\_cfa\_offset 16

.cfi\_offset 6, -16

movq %rsp, %rbp

.cfi\_def\_cfa\_register 6

subq $48, %rsp

pxor %xmm0, %xmm0

movsd %xmm0, -32(%rbp)

movsd .LC5(%rip), %xmm0

movsd %xmm0, -24(%rbp)

movsd .LC10(%rip), %xmm0

movsd %xmm0, -16(%rbp)

pxor %xmm0, %xmm0

movsd %xmm0, -8(%rbp)

movsd .LC11(%rip), %xmm1

movsd -32(%rbp), %xmm2

movsd -16(%rbp), %xmm0

movq -24(%rbp), %rax

movapd %xmm2, %xmm3

movapd %xmm1, %xmm2

movapd %xmm0, %xmm1

movq %rax, -40(%rbp)

movsd -40(%rbp), %xmm0

call simps

movq %xmm0, %rax

movq %rax, -8(%rbp)

movq -8(%rbp), %rax

movq %rax, -40(%rbp)

movsd -40(%rbp), %xmm0

leaq .LC12(%rip), %rdi

movl $1, %eax

call printf@PLT

movl $0, %eax

leave

.cfi\_def\_cfa 7, 8

ret

.cfi\_endproc

.LFE8:

.size main, .-main

.section .rodata

.align 16

.LC0:

.long 0

.long -2147483648

.long 0

.long 0

.align 8

.LC1:

.long 0

.long 1073741824

.align 8

.LC3:

.long 0

.long 1074790400

.align 8

.LC4:

.long 0

.long 1074266112

.align 8

.LC5:

.long 0

.long 1072693248

.align 8

.LC6:

.long 0

.long 1075576832

.align 8

.LC7:

.long 0

.long 1076625408

.align 8

.LC8:

.long 0

.long 1076887552

.align 8

.LC9:

.long 0

.long 1076756480

.align 8

.LC10:

.long 0

.long 1075970048

.align 8

.LC11:

.long 2696277389

.long 1051772663

.ident "GCC: (Ubuntu 7.4.0-1ubuntu1~18.04.1) 7.4.0"

.section .note.GNU-stack,"",@progbits