# 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

|  |  |  |
| --- | --- | --- |
| Студент гр. 7304 |  | Кошманов Н.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 (c 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.

При настройке параметров (ключей) запуска программы автоматизации следует задать корректное значение числа внешних связей 2\* анализируемой программы (по умолчанию задается 5), совпадающее с используемым при ручном расчете.

Результаты расчетов представить в виде таблиц с текстовыми комментариями:

* 1. Паскаль. Ручной расчет:
     1. Измеримые характеристики,
     2. Расчетные характеристики
  2. Паскаль. Программный расчет:

а) Измеримые характеристики, б) Расчетные характеристики

* 1. Си. Ручной расчет:
     1. Измеримые характеристики,
     2. Расчетные характеристики
  2. Си. Программный расчет:
     1. Измеримые характеристики,
     2. Расчетные характеристики
  3. Ассемблер. Ручной расчет:
     1. Измеримые характеристики,
     2. Расчетные характеристики
  4. Сводная таблица расчетов для трех языков.

# Ход работы

1. Вариант – 8, **Умножение матриц**.

Исходный код программы представлен в приложении А. В данной реализации исключены операторы, обеспечивающие интерфейс с пользователем и выдачу текстовых сообщений. Исправленная версия программы представлена в приложении Б.

1. Ручной расчет метрик программы на Pascal

Для программы из Приложения Б был произведен подсчет операторов и операндов в программе, а также расчет измеримых и расчетных характеристик программы. Результаты представлены в табл. 1, 2.

Таблица 1 — Операторы и операнды программы на Pascal

|  |  |  |  |
| --- | --- | --- | --- |
| **Оператор** | **Количество** | **Операнд** | **Количество** |
| **square** | 2 | **i** | 12 |
| **get\_data** | 2 | **l** | 8 |
| **if…then** | 1 | **k** | 13 |
| **<>** | 1 | **x** | 10 |
| **begin…end** | 7 | **a** | 8 |
| **for to do** | 6 | **g** | 6 |
| **+** | 2 | **j** | 7 |
| **\*** | 4 | **ncol** | 8 |
| **;** | 20 | **nrow** | 7 |

|  |  |  |  |
| --- | --- | --- | --- |
| **-** | 1 | **y** | 6 |
| **()** | 2 | **1** | 8 |
| **:=** | 16 | **2** | 2 |
| **[]** | 16 | **0** | 2 |

Таблица 2 — Измеримые и расчетные характеристики для Pascal (ручной подсчет)

|  |  |  |
| --- | --- | --- |
| **Метрика** | **Описание** | **Значение** |
| **η1** | Число простых (уникальных) операторов | 13 |
| **η2** | Число простых (уникальных) операндов | 13 |
| **N1** | Общее число всех операторов | 80 |
| **N2** | Общее число всех операндов | 97 |
| **η** | Словарь, **η1 + η2** | 26 |
| **N** | Опытная длина, **N1 + N2** | 177 |
| **Nтеор** | Теоретическая длина, **η1log2η1 + η2log2η2** | 96,211 |
| **V** | Объем, **N log2η** | 831,978 |
| **V\*** | Потенциальный объем, (**η2\* + 2) log2(η2\* + 2), η2\*=6** | 24 |
| **L** | Уровень, **V\*/ V** | 0,0288 |
| **I** | Интеллектуальное содержание, **2/η1\*η2/N2\*N\* log2(η)** | 17,15 |
| **E** | Работа по программированию, **V/L** | 28841,12 |
| **Ť** | Время программирования, **E/S, S=10** | 2884,112 |
| **λ** | Уровень языка, **L\*V\*** | 0,692326 |
| **B** | Количество ошибок, **V\*/L/1000** | 1 |

1. Программный расчет метрик программы на Pascal Результаты представлены в табл. 3.

Таблица 3 — Измеримые и расчетные характеристики для Pascal (программный подсчет)

|  |  |  |
| --- | --- | --- |
| **Метрика** | **Описание** | **Значение** |
| **η1** | Число простых (уникальных) операторов | 14 |
| **η2** | Число простых (уникальных) операндов | 23 |
| **N1** | Общее число всех операторов | 61 |
| **N2** | Общее число всех операндов | 120 |
| **η** | Словарь, **η1 + η2** | 37 |
| **N** | Опытная длина, **N1 + N2** | 181 |
| **Nтеор** | Теоретическая длина, **η1log2η1 + η2log2η2** | 157,345 |
| **V** | Объем, **N log2η** | 942,911 |
| **V\*** | Потенциальный объем, (**η2\* + 2) log2(η2\* + 2), η2\*=6** | 24 |
| **L** | Уровень, **V\*/ V** | 0,02545 |
| **I** | Интеллектуальное содержание, **2/η1\*η2/N2\*N\* log2(η)** | 25,8178 |
| **E** | Работа по программированию, **V/L** | 37045,1 |
| **Ť** | Время программирования, **E/S, S=10** | 3704,51 |
| **λ** | Уровень языка, **L\*V\*** | 0.610874 |
| **B** | Количество ошибок, **V\*/L/1000** | 1 |

1. Ручной расчет метрик программы на C

Для программы из Приложения Б была написана программа на языке С, был произведен подсчет операторов и операндов в программе на С, а также расчет измеримых и расчетных характеристик программы. Результаты представлены в табл. 4, 5

Таблица 4 — Операторы и операнды программы на С

|  |  |  |  |
| --- | --- | --- | --- |
| **Оператор** | **Количество** | **Операнд** | **Количество** |
| **;** | 23 | i | 20 |

|  |  |  |  |
| --- | --- | --- | --- |
| **=** | 18 | l | 10 |
| **[] or [][]** | 16 | k | 15 |
| **()** | 11 | x | 11 |
| **{}** | 9 | a | 7 |
| **++** | 6 | g | 6 |
| **for** | 6 | j | 5 |
| **<** | 6 | ncol | 7 |
| **\*** | 4 | nrow | 8 |
| **pointer\*** | 2 | y | 7 |
| **+** | 4 | 1 | 5 |
| **square** | 2 | 2 | 1 |
| **get\_data** | 2 | 0 | 9 |
| **!=** | 1 |  |  |
| **-** | 1 |  |  |
| **<=** | 1 |  |  |
| **if** | 1 |  |  |

Таблица 5 — Измеримые и расчетные характеристики для С (ручной подсчет)

|  |  |  |
| --- | --- | --- |
| **Метрика** | **Описание** | **Значение** |
| **η1** | Число простых (уникальных) операторов | 17 |
| **η2** | Число простых (уникальных) операндов | 13 |
| **N1** | Общее число всех операторов | 113 |
| **N2** | Общее число всех операндов | 111 |
| **η** | Словарь, **η1 + η2** | 30 |
| **N** | Опытная длина, **N1 + N2** | 224 |
| **Nтеор** | Теоретическая длина, **η1log2η1 + η2log2η2** | 117,5926 |
| **V** | Объем, **N log2η** | 1099,143 |
| **V\*** | Потенциальный объем, (**η2\* + 2) log2(η2\* + 2), η2\*=6** | 24 |

|  |  |  |
| --- | --- | --- |
| **L** | Уровень, **V\*/ V** | 0,0218 |
| **I** | Интеллектуальное содержание, **2/η1\*η2/N2\*N\* log2(η)** | 15,144 |
| **E** | Работа по программированию, **V/L** | 50338,18 |
| **Ť** | Время программирования, **E/S, S=10** | 5033,818 |
| **λ** | Уровень языка, **L\*V\*** | 0,52404 |
| **B** | Количество ошибок, **V\*/L/1000** | 2 |

1. Программный расчет метрик программы на C Результаты представлены в табл. 6.

Таблица 6 — Измеримые и расчетные характеристики для C (программный подсчет)

|  |  |  |
| --- | --- | --- |
| **Метрика** | **Описание** | **Значение** |
| **η1** | Число простых (уникальных) операторов | 20 |
| **η2** | Число простых (уникальных) операндов | 17 |
| **N1** | Общее число всех операторов | 122 |
| **N2** | Общее число всех операндов | 119 |
| **η** | Словарь, **η1 + η2** | 37 |
| **N** | Опытная длина, **N1 + N2** | 241 |
| **Nтеор** | Теоретическая длина, **η1log2η1 + η2log2η2** | 155,925 |
| **V** | Объем, **N log2η** | 1255,48 |
| **V\*** | Потенциальный объем, (**η2\* + 2) log2(η2\* + 2), η2\*=6** | 24 |
| **L** | Уровень, **V\*/ V** | 0,01911 |
| **I** | Интеллектуальное содержание, **2/η1\*η2/N2\*N\* log2(η)** | 17,934 |
| **E** | Работа по программированию, **V/L** | 65676,1 |
| **Ť** | Время программирования, **E/S, S=10** | 6567,61 |
| **λ** | Уровень языка, **L\*V\*** | 0.458789 |
| **B** | Количество ошибок, **V\*/L/1000** | 2 |

1. Ручной расчет метрик программы на Assembly

Для программы из Приложения В был создан код на языке Assembly, используя gcc -S program.c. Был произведен подсчет операторов и операндов в программе на Assembly, а также расчет измеримых и расчетных характеристик программы. Результаты представлены в табл. 7, 7

Таблица 7 — Операторы и операнды программы на Assembly

|  |  |  |  |
| --- | --- | --- | --- |
| **Оператор** | **Количество** | **Операнд** | **Количество** |
| **addl** | 8 | **$0** | 6 |
| **addq** | 37 | $1 | 10 |
| **addsd** | 2 | **$3** | 12 |
| **call** | 3 | **$5** | 1 |
| **cltq** | 15 | **$432** | 1 |
| **cmpl** | 7 | **%eax** | 60 |
| **cvtsi2sd** | 2 | **%ecx** | 4 |
| **get\_data** | 1 | **%edi** | 2 |
| **get\_data:** | 1 | **%edx** | 2 |
| **je** | 2 | **%r8d** | 4 |
| **jl** | 5 | **%r9d** | 2 |
| **jle** | 1 | **%rax** | 103 |
| **jmp** | 6 | **%rbp** | 8 |
| **leal** | 1 | **%rcx** | 3 |
| **leaq** | 12 | **%rdi** | 6 |
| **leave** | 1 | **%rdx** | 66 |
| **main:** | 1 | **%rsi** | 4 |
| **movl** | 54 | **%rsp** | 4 |
| **movq** | 52 | **%xmm0** | 23 |
| **movsd** | 17 | **%xmm1** | 6 |

|  |  |  |  |
| --- | --- | --- | --- |
| **movslq** | 11 | **%xmm2** | 4 |
| **mulsd** | 3 | **(%rax%rax)** | 1 |
| **nop** | 2 | **(%rax)** | 6 |
| **popq** | 2 | **(%rcx%rax8)** | 1 |
| **pushq** | 3 | **(%rdx%rax)** | 1 |
| **pxor** | 2 | **(%rdx%rax8)** | 9 |
| **ret** | 3 | **-12(%rbp)** | 10 |
| **salq** | 11 | **-16(%rbp)** | 15 |
| **square** | 1 | **-224(%rbp)** | 2 |
| **square:** | 1 | **-24(%rbp)** | 8 |
| **subl** | 1 | **-304(%rbp)** | 1 |
| **subq** | 1 | **-32(%rbp)** | 4 |
| **xorl** | 1 | **-36(%rbp)** | 2 |
| **xorq** | 1 | **-384(%rbp)** | 2 |
|  |  | **-4(%rbp)** | 10 |
|  |  | **-40(%rbp)** | 8 |
|  |  | **-416(%rbp)** | 1 |
|  |  | **-420(%rbp)** | 3 |
|  |  | **-424(%rbp)** | 3 |
|  |  | **-48(%rbp)** | 4 |
|  |  | **-52(%rbp)** | 3 |
|  |  | **-56(%rbp)** | 2 |
|  |  | **-8(%rbp)** | 16 |
|  |  | **.LC0(%rip)** | 1 |
|  |  | **0(%rax8)** | 1 |

Таблица 8 — Измеримые и расчетные характеристики для Assembly (ручной подсчет)

|  |  |  |
| --- | --- | --- |
| **Метрика** | **Описание** | **Значение** |
| **η1** | Число простых (уникальных) операторов | 34 |
| **η2** | Число простых (уникальных) операндов | 45 |
| **N1** | Общее число всех операторов | 271 |
| **N2** | Общее число всех операндов | 445 |
| **η** | Словарь, **η1 + η2** | 79 |
| **N** | Опытная длина, **N1 + N2** | 716 |
| **Nтеор** | Теоретическая длина, **η1log2η1 + η2log2η2** | 420,1071 |
| **V** | Объем, **N log2η** | 4513,507 |
| **V\*** | Потенциальный объем, (**η2\* + 2) log2(η2\* + 2), η2\*=6** | 24 |
| **L** | Уровень, **V\*/ V** | 0,005317 |
| **I** | Интеллектуальное содержание, **2/η1\*η2/N2\*N\* log2(η)** | 26,84836 |
| **E** | Работа по программированию, **V/L** | 848822,7 |
| **Ť** | Время программирования, **E/S, S=10** | 84882,27 |
| **λ** | Уровень языка, **L\*V\*** | 0,1276 |
| **B** | Количество ошибок, **V\*/L/1000** | 5 |

1. Сводная таблица результатов

Для языков Pascal, C, Assembly составлена сводная таблица 9 по измеримым и расчетным характеристикам.

Таблица 9 — Сводная таблица расчетов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Pascal** | | **C** | | **Assembly** |
| **Ручной** | **Программный** | **Ручной** | **Программный** | **Ручной** |
| **η1** | 13 | 14 | 17 | 20 | 34 |
| **η2** | 13 | 23 | 13 | 17 | 45 |
| **N1** | 80 | 61 | 113 | 122 | 271 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **N2** | 97 | 120 | 111 | 119 | 445 |
| **η** | 26 | 37 | 30 | 37 | 79 |
| **N** | 177 | 181 | 224 | 241 | 716 |
| **Nтеор** | 96,211 | 157,345 | 117,5926 | 155,925 | 420,1071 |
| **V** | 831,978 | 942,911 | 1099,143 | 1255,48 | 4513,507 |
| **V\*** | 24 | 24 | 24 | 24 | 24 |
| **L** | 0,0288 | 0,02545 | 0,0218 | 0,01911 | 0,005317 |
| **I** | 17,15 | 25,8178 | 15,144 | 17,934 | 26,84836 |
| **E** | 28841,12 | 37045,1 | 50338,18 | 65676,1 | 848822,7 |
| **Ť** | 2884,112 | 3704,51 | 5033,818 | 6567,61 | 84882,27 |
| **λ** | 0,692326 | 0.610874 | 0,52404 | 0.458789 | 0,1276 |
| **B** | 1 | 1 | 2 | 2 | 5 |

# 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 на Pascal**

program matr1;

{ pascal program to perform matrix multiplication }

const rmax = 9;

cmax = 3;

type ary = array[1..rmax] of real; arys = array[1..cmax] of real;

ary2 = array[1..rmax,1..cmax] of real; ary2s = array[1..cmax,1..cmax] of real;

var y : ary;

g : arys;

x : ary2;

a : ary2s;

nrow,ncol : integer;

procedure get\_data(var x: ary2;

var y: ary;

var nrow,ncol: integer);

{ get the values for nrow, ncol, and arrays x,y } var i,j : integer;

begin

nrow:=5; {эти значения не следует считать } ncol:=3; { фиксированными - они могут изменяться} for i:=1 to nrow do

begin

x[i,1]:=1;

for j:=2 to ncol do x[i,j]:=i\*x[i,j-1];

y[i]:=2\*i end

end; { procedure get\_data } procedure write\_data;

{ print out the answeres } var

i,j : integer;

begin

ClrScr; writeln;

writeln(' X Y'); for i:=1 to nrow do

begin

for j:=1 to ncol do write(x[i,j]:7:1,' ');

writeln(':',y[i]:7:1) end;

writeln(' A G'); for i:=1 to ncol do

begin

for j:=1 to ncol do

write(a[i,j]:7:1,' ');

writeln(':',g[i]:7:1) end

end; { write\_data }

procedure square(x: ary2;

y: ary; var a: ary2s; var g: arys;

nrow,ncol: integer);

{ matrix multiplication routine }

{ a= transpose x times x }

{ g= y times x }

var

i,k,l : integer;

begin { square } for k:=1 to ncol do

begin

for l:=1 to k do begin

a[k,l]:=0;

for i:=1 to nrow do begin

a[k,l]:=a[k,l]+x[i,l]\*x[i,k]; if k<>l then a[l,k]:=a[k,l]

end

end; { l-loop }

g[k]:=0;

for i:=1 to nrow do g[k]:=g[k]+y[i]\*x[i,k]

end { k-loop } end; { square }

begin { MAIN program } get\_data(x,y,nrow,ncol); square(x,y,a,g,nrow,ncol); write\_data

end.

program matr1; const

rmax = 9;

cmax = 3;

# ПРИЛОЖЕНИЕ Б

**Измененный код на Pascal**

type

ary = array[1..rmax] of real; arys = array[1..cmax] of real;

ary2 = array[1..rmax, 1..cmax] of real; ary2s = array[1..cmax, 1..cmax] of real;

var

y: ary; g: arys; x: ary2; a: ary2s;

nrow, ncol: integer;

procedure get\_data(var x: ary2; var y: ary; var nrow, ncol: integer); var

i, j: integer;

begin

nrow := 5;

ncol := 3;

for i := 1 to nrow do begin

x[i, 1] := 1;

for j := 2 to ncol do

x[i, j] := i \* x[i, j - 1]; y[i] := 2 \* i;

end; end;

procedure square(x: ary2; y: ary; var a: ary2s; var g: arys; nrow, ncol: integer);

var

i, k, l: integer;

begin

for k := 1 to ncol do begin

for l := 1 to k do begin

a[k, l] := 0;

for i := 1 to nrow do begin

a[k, l] := a[k, l] + x[i, l] \* x[i, k]; if k <> l then

a[l, k] := a[k, l]; end;

end;

g[k] := 0;

for i := 1 to nrow do

g[k] := g[k] + y[i] \* x[i, k];

end; end;

begin

get\_data(x, y, nrow, ncol); square(x, y, a, g, nrow, ncol);

end.

#include <stdio.h> #define rmax 9

#define cmax 3

# ПРИЛОЖЕНИЕ В

**Исходный код на C**

typedef double ary[rmax]; typedef double arys[cmax]; typedef double ary2[rmax][cmax];

typedef double ary2s[cmax][cmax]; typedef double\* pointer;

typedef double\*\* dpointer;

void get\_data(ary2 x, ary y, int nrow, int ncol) { for (int i = 0; i < nrow; i++) {

x[i][0] = 1;

for (int j = 1; j < ncol; j++) { x[i][j] = (i + 1) \* x[i][j - 1];

}

y[i] = 2 \* (i + 1);

}

}

void square(ary2 x, double\* y, ary2s a, double\* g, int nrow, int ncol) { for (int k = 0; k < ncol; k++) {

for (int l = 0; l <= k; l++) { a[k][l] = 0;

for (int i = 0; i < nrow; i++) {

a[k][l] = a[k][l] + x[i][l] \* x[i][k]; if (k != l)

a[l][k] = a[k][l];

}

}

g[k] = 0;

for (int i = 0; i < nrow; i++) { g[k] = g[k] + y[i] \* x[i][k];

}

}

}

int main() {

int nrow = 5; int ncol = 3;

ary2 x; ary y;

arys g; ary2s a;

get\_data(x, y, nrow, ncol); square(x, y, a, g, nrow, ncol);

return 0;

}

# ПРИЛОЖЕНИЕ Г

**Исходный код на Assembly**

get\_data:

pushq %rbp

movq %rsp, %rbp movq %rdi, -24(%rbp) movq %rsi, -32(%rbp) movl %edx, -36(%rbp) movl %ecx, -40(%rbp) movl $0, -8(%rbp) jmp .L2

movl -8(%rbp), %eax movslq %eax, %rdx movq %rdx, %rax addq %rax, %rax addq %rdx, %rax salq $3, %rax

movq %rax, %rdx movq -24(%rbp), %rax addq %rdx, %rax

movsd .LC0(%rip), %xmm0 movsd %xmm0, (%rax) movl $1, -4(%rbp)

jmp .L3

movl -8(%rbp), %eax addl $1, %eax

cvtsi2sd %eax, %xmm1 movl -8(%rbp), %eax movslq %eax, %rdx

movq %rdx, %rax addq %rax, %rax addq %rdx, %rax salq $3, %rax movq %rax, %rdx

movq -24(%rbp), %rax addq %rax, %rdx movl -4(%rbp), %eax subl $1, %eax

cltq

movsd (%rdx,%rax,8), %xmm0 movl -8(%rbp), %eax movslq %eax, %rdx

movq %rdx, %rax addq %rax, %rax addq %rdx, %rax salq $3, %rax movq %rax, %rdx

movq -24(%rbp), %rax addq %rax, %rdx mulsd %xmm1, %xmm0 movl -4(%rbp), %eax cltq

movsd %xmm0, (%rdx,%rax,8) addl $1, -4(%rbp)

movl -4(%rbp), %eax cmpl -40(%rbp), %eax jl .L4

movl -8(%rbp), %eax addl $1, %eax

leal (%rax,%rax), %ecx

movl -8(%rbp), %eax cltq

leaq 0(,%rax,8), %rdx movq -32(%rbp), %rax addq %rdx, %rax cvtsi2sd %ecx, %xmm0 movsd %xmm0, (%rax) addl $1, -8(%rbp)

movl -8(%rbp), %eax cmpl -36(%rbp), %eax jl .L5

nop

popq %rbp ret

square:

pushq %rbp

movq %rsp, %rbp movq %rdi, -24(%rbp) movq %rsi, -32(%rbp) movq %rdx, -40(%rbp) movq %rcx, -48(%rbp) movl %r8d, -52(%rbp) movl %r9d, -56(%rbp) movl $0, -16(%rbp) jmp .L7

movl $0, -12(%rbp) jmp .L8

movl -16(%rbp), %eax movslq %eax, %rdx movq %rdx, %rax addq %rax, %rax addq %rdx, %rax salq $3, %rax

movq %rax, %rdx movq -40(%rbp), %rax addq %rax, %rdx movl -12(%rbp), %eax cltq

pxor %xmm0, %xmm0

movsd %xmm0, (%rdx,%rax,8) movl $0, -8(%rbp)

jmp .L9

movl -16(%rbp), %eax movslq %eax, %rdx movq %rdx, %rax addq %rax, %rax addq %rdx, %rax salq $3, %rax

movq %rax, %rdx movq -40(%rbp), %rax addq %rax, %rdx movl -12(%rbp), %eax cltq

movsd (%rdx,%rax,8), %xmm1 movl -8(%rbp), %eax movslq %eax, %rdx

movq %rdx, %rax addq %rax, %rax addq %rdx, %rax salq $3, %rax movq %rax, %rdx

movq -24(%rbp), %rax addq %rax, %rdx movl -12(%rbp), %eax cltq

movsd (%rdx,%rax,8), %xmm2 movl -8(%rbp), %eax movslq %eax, %rdx

movq %rdx, %rax addq %rax, %rax addq %rdx, %rax salq $3, %rax movq %rax, %rdx

movq -24(%rbp), %rax addq %rax, %rdx movl -16(%rbp), %eax cltq

movsd (%rdx,%rax,8), %xmm0 mulsd %xmm2, %xmm0

movl -16(%rbp), %eax movslq %eax, %rdx movq %rdx, %rax addq %rax, %rax addq %rdx, %rax salq $3, %rax

movq %rax, %rdx movq -40(%rbp), %rax addq %rax, %rdx addsd %xmm1, %xmm0 movl -12(%rbp), %eax cltq

movsd %xmm0, (%rdx,%rax,8) movl -16(%rbp), %eax

cmpl -12(%rbp), %eax je .L10

movl -16(%rbp), %eax movslq %eax, %rdx movq %rdx, %rax addq %rax, %rax addq %rdx, %rax salq $3, %rax

movq %rax, %rdx movq -40(%rbp), %rax

leaq (%rdx,%rax), %rcx movl -12(%rbp), %eax movslq %eax, %rdx

movq %rdx, %rax addq %rax, %rax addq %rdx, %rax salq $3, %rax movq %rax, %rdx

movq -40(%rbp), %rax addq %rax, %rdx movl -12(%rbp), %eax cltq

movsd (%rcx,%rax,8), %xmm0 movl -16(%rbp), %eax cltq

movsd %xmm0, (%rdx,%rax,8) addl $1, -8(%rbp)

movl -8(%rbp), %eax cmpl -52(%rbp), %eax

main:

jl .L11

addl $1, -12(%rbp) movl -12(%rbp), %eax cmpl -16(%rbp), %eax jle .L12

movl -16(%rbp), %eax cltq

leaq 0(,%rax,8), %rdx movq -48(%rbp), %rax addq %rdx, %rax

pxor %xmm0, %xmm0 movsd %xmm0, (%rax) movl $0, -4(%rbp) jmp .L13

movl -16(%rbp), %eax cltq

leaq 0(,%rax,8), %rdx movq -48(%rbp), %rax addq %rdx, %rax movsd (%rax), %xmm1 movl -4(%rbp), %eax cltq

leaq 0(,%rax,8), %rdx movq -32(%rbp), %rax addq %rdx, %rax movsd (%rax), %xmm2 movl -4(%rbp), %eax movslq %eax, %rdx

movq %rdx, %rax addq %rax, %rax addq %rdx, %rax salq $3, %rax movq %rax, %rdx

movq -24(%rbp), %rax addq %rax, %rdx movl -16(%rbp), %eax cltq

movsd (%rdx,%rax,8), %xmm0 mulsd %xmm2, %xmm0

movl -16(%rbp), %eax cltq

leaq 0(,%rax,8), %rdx movq -48(%rbp), %rax addq %rdx, %rax addsd %xmm1, %xmm0 movsd %xmm0, (%rax) addl $1, -4(%rbp) movl -4(%rbp), %eax cmpl -52(%rbp), %eax jl .L14

addl $1, -16(%rbp) movl -16(%rbp), %eax cmpl -56(%rbp), %eax jl .L15

nop

popq %rbp ret

pushq %rbp

movq %rsp, %rbp subq $432, %rsp

movq %fs:40, %rax movq %rax, -8(%rbp) xorl %eax, %eax movl $5, -424(%rbp) movl $3, -420(%rbp)

movl -420(%rbp), %ecx movl -424(%rbp), %edx leaq -384(%rbp), %rsi leaq -224(%rbp), %rax movq %rax, %rdi

call get\_data

movl -420(%rbp), %r8d movl -424(%rbp), %edi leaq -416(%rbp), %rcx leaq -304(%rbp), %rdx leaq -384(%rbp), %rsi leaq -224(%rbp), %rax movl %r8d, %r9d

movl %edi, %r8d movq %rax, %rdi call square movl $0, %eax

movq -8(%rbp), %rdi xorq %fs:40, %rdi je .L18

call stack\_chk\_fail@PLT leave

ret