**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 программ по метрикам Холстеда

|  |  |  |
| --- | --- | --- |
| Студент гр. 8304 |  | Алтухов А.Д. |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

2022

**Цель работы.**

Изучить метрические характеристики качества разработки программ на основе метрик Холстеда для программ на Pascal, C и ассемблере.

**Ход выполнения.**

Был выбран вариант 1 — численное интегрирование методом Симпсона. Исходный код представлен в Приложении А. Для этой программы был произведен вручную расчет операторов и операндов. Результат представлен в таблице 1.

Таблица 1. Ручной расчет операторов и операндов в программе на Pascal.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Оператор | Количество | № | Операнд | Количество |
| 1 | ; | 28 | 1 | tol | 2 |
| 2 | := | 20 | 2 | sum | 7 |
| 3 | () | 22 | 3 | upper | 6 |
| 4 | begin … end | 5 | 4 | lower | 8 |
| 5 | repeat … until | 1 | 5 | x | 4 |
| 6 | for ... to … do | 1 | 6 | fx | 1 |
| 7 | fx | 8 | 7 | dfx | 1 |
| 8 | dfx | 2 | 8 | i | 2 |
| 9 | simps | 1 | 9 | delta\_x | 7 |
| 10 | abs | 2 | 10 | even\_sum | 4 |
| 11 | and | 1 | 11 | odd\_sum | 7 |
| 12 | + | 9 | 12 | end\_sum | 3 |
| 13 | - | 8 | 13 | end\_cor | 2 |
| 14 | \* | 11 | 14 | sum1 | 3 |
| 15 | / | 7 | 15 | pieces | 6 |
| 16 | div | 1 | 16 | 1.0E-6 | 1 |
| 17 | <> | 1 | 17 | 2 | 7 |
| 18 | <= | 1 | 18 | 0 | 2 |
| 19 | function fx | 1 | 19 | 4 | 1 |
| 20 | function dfx | 1 | 20 | 3 | 1 |
| 21 | procedure simps | 1 | 21 | 1 | 3 |
| 22 | exp | 2 | 22 | 7 | 1 |
|  |  |  | 23 | 9 | 1 |
|  |  |  | 24 | 14 | 1 |
|  |  |  | 25 | 15 | 1 |
|  |  |  | 26 | 16 | 1 |

Далее был произведен расчет измеримых характеристик этой программы. Полученный результат представлен в таблице 2.

Таблица 2. Ручное определение измеримых характеристик программы на Pascal.

|  |  |  |
| --- | --- | --- |
| Характеристика | Формула | Значение |
| Число уникальных операторов | η1 | 22 |
| Число уникальных операндов | η2 | 26 |
| Число всех операторов | N1 | 134 |
| Число всех операндов | N2 | 83 |
| Словарь программы | η1 = η1 + η2 | 48 |
| Длина программы | N = N1 + N2 | 217 |

Далее были произведены расчеты для получения расчетных характеристик. Результаты представлены в таблице 3. Для расчетов значение коэффициента Стауда S принято 10, значение 𝜂2∗ принято 4, так как процедура simps принимает 4 параметра, из которых в один записывается возвращаемое значение.

Таблица 3. Ручное определение расчетных характеристик программы на Pascal.

|  |  |  |
| --- | --- | --- |
| Характеристика | Формула | Значение |
| Теоретическая оценка длины программы | Ň=η1log2η1+η2log2η2 | 220,32 |
| Реальный объём | 𝑉 = 𝑁 log2η2 | 1020 |
| Потенциальный объём | V\*=(2+η2\*)log2(2+η2\*) | 15,51 |
| Уровень программы | *L=V\*/V* | 0,015 |
| Интеллектуальное содержание программы | *I=2η2/(η1N2)(N1+N2)log2(η1+η2)* | 34,51 |
| Работа программиста | *E=V/L* | 68000 |
| Время программирования | *T=E/S* | 6800 |
| Уровень языка | 𝜆 =*LV\** | 0,233 |
| Ожидаемое число ошибок | B=(*V\**)2/(1000𝜆) | 1,03 |

С помощью программы автоматизации расчета метрик Холстеда были подсчитаны операторы и операнды в программе на Pascal, определены измеримые характеристики программы на Pascal, определены расчетные характеристики программы на Pascal.

Таблица 4. Программный расчет операторов и операндов в программе на Pascal.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Оператор | Количество | № | Операнд | Количество |
| 1 | () | 23 | 1 | 0 | 2 |
| 2 | \* | 11 | 2 | 1 | 3 |
| 3 | + | 9 | 3 | 1.0E-6 | 1 |
| 4 | - | 8 | 4 | 14 | 1 |
| 5 | / | 8 | 5 | 15 | 1 |
| 6 | ; | 44 | 6 | 16 | 1 |
| 7 | <= | 1 | 7 | 2 | 7 |
| 8 | <> | 1 | 8 | 3 | 1 |
| 9 | = | 20 | 9 | 4 | 1 |
| 10 | abs | 2 | 10 | 7 | 1 |
| 11 | and | 1 | 11 | 9 | 1 |
| 12 | const | 1 | 12 | delta\_x | 8 |
| 13 | dfx | 3 | 13 | dfx | 1 |
| 14 | exp | 2 | 14 | end\_cor | 3 |
| 15 | for | 1 | 15 | end\_sum | 4 |
| 16 | fx | 5 | 16 | even\_sum | 5 |
| 17 | program | 1 | 17 | fx | 1 |
| 18 | real | 2 | 18 | i | 2 |
| 19 | repeat | 1 | 19 | lower | 10 |
| 20 | simps | 2 | 20 | odd\_sum | 8 |
|  |  |  | 21 | pieces | 7 |
|  |  |  | 22 | simp1 | 1 |
|  |  |  | 23 | sum | 9 |
|  |  |  | 24 | sum1 | 4 |
|  |  |  | 25 | tol | 4 |
|  |  |  | 26 | upper | 8 |
|  |  |  | 27 | x | 7 |

Таблица 5. Программное определение измеримых характеристик программы на Pascal.

|  |  |  |
| --- | --- | --- |
| Характеристика | Формула | Значение |
| Число уникальных операторов | η1 | 20 |
| Число уникальных операндов | η2 | 27 |
| Число всех операторов | N1 | 146 |
| Число всех операндов | N2 | 102 |
| Словарь программы | η1 = η1 + η2 | 47 |
| Длина программы | N = N1 + N2 | 248 |

Таблица 6. Программное определение расчетных характеристик программы на Pascal.

|  |  |  |
| --- | --- | --- |
| Характеристика | Формула | Значение |
| Теоретическая оценка длины программы | Ň=η1log2η1+η2log2η2 | 214.821 |
| Реальный объём | 𝑉 = 𝑁 log2η2 | 1377,54 |
| Потенциальный объём | V\*=(2+η2\*)log2(2+η2\*) | 15,51 |
| Уровень программы | *L=V\*/V* | 0.011 |
| Интеллектуальное содержание программы | *I=2η2/(η1N2)(N1+N2)log2(η1+η2)* | 36,46 |
| Работа программиста | *E=V/L* | 122349 |
| Время программирования | *T=E/S* | 12234,9 |
| Уровень языка | 𝜆 =*LV\** | 0,175 |
| Ожидаемое число ошибок | B=(*V\**)2/(1000𝜆) | 0,82 |

На основе программы на языке Pascal была написана аналогичная программа на языке С.

Результат работы программы на С представлен в приложении Б. Код программы представлен в приложении В. Для этой программы был произведен ручной расчет операторов и операндов. Результат представлен в таблице 7.

Таблица 7. Ручной расчет операторов и операндов в программе на C.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Оператор | Количество | № | Операнд | Количество |
| 1 | ; | 27 | 1 | tol | 2 |
| 2 | *=* | 19 | 2 | sum | 8 |
| 3 | () | 22 | 3 | upper | 6 |
| 4 | {} | 6 | 4 | lower | 8 |
| 5 | + | 9 | 5 | x | 4 |
| 6 | - | 8 | 6 | i | 2 |
| 7 | \* | 17 | 7 | delta\_x | 7 |
| 8 | / | 8 | 8 | even\_sum | 4 |
| 9 | ++ | 1 | 9 | odd\_sum | 7 |
| 10 | & | 1 | 10 | end\_sum | 3 |
| 11 | && | 1 | 11 | end\_cor | 2 |
| 12 | <= | 2 | 12 | sum1 | 3 |
| 13 | != | 1 | 13 | pieces | 6 |
| 14 | fx | 4 | 14 | 1.0E-6 | 1 |
| 15 | dfx | 2 | 15 | 2 | 7 |
| 16 | exp | 2 | 16 | 0 | 4 |
| 17 | fabs | 2 | 17 | 4 | 1 |
| 18 | simps | 1 | 18 | 3 | 1 |
| 19 | do … while | 1 | 19 | 1 | 3 |
| 20 | for | 1 | 20 | 7 | 1 |
| 21 | return | 3 | 21 | 9 | 1 |
| 22 | double fx | 1 | 22 | 14 | 1 |
| 23 | double dfx | 1 | 23 | 15 | 1 |
| 24 | double simps | 1 | 24 | 16 | 1 |
| 25 | int main | 1 |  |  |  |

Далее был произведен расчет измеримых характеристик этой программы. Полученный результат представлен в таблице 8.

Таблица 8. Ручное определение измеримых характеристик программы на C.

|  |  |  |
| --- | --- | --- |
| Характеристика | Формула | Значение |
| Число уникальных операторов | η1 | 25 |
| Число уникальных операндов | η2 | 24 |
| Число всех операторов | N1 | 142 |
| Число всех операндов | N2 | 84 |
| Словарь программы | η= η1 + η2 | 49 |
| Длина программы | N = N1 + N2 | 226 |

Далее были произведены расчеты для получения расчетных характеристик. Результаты представлены в таблице 9. Для расчетов значение коэффициента Стауда S принято 10, значение 𝜂2∗ принято 4, так как функция simps принимает 4 параметра, из которых в один записывается возвращаемое значение.

Таблица 9. Ручное определение расчетных характеристик программы на C.

|  |  |  |
| --- | --- | --- |
| Характеристика | Формула | Значение |
| Теоретическая оценка длины программы | Ň=η1log2η1+η2log2η2 | 226,14 |
| Реальный объём | 𝑉 = 𝑁 log2η | 1268,92 |
| Потенциальный объём | V\*=(2+η2\*)log2(2+η2\*) | 15,51 |
| Уровень программы | *L=V\*/V* | 0,012 |
| Интеллектуальное содержание программы | *I=2η2/(η1N2)(N1+N2)log2(η1+η2)* | 29 |
| Работа программиста | *E=V/L* | 105743,3 |
| Время программирования | *T=E/S* | 10574,33 |
| Уровень языка | 𝜆 =*LV\** | 0,186 |
| Ожидаемое число ошибок | B=(*V\**)2/(1000𝜆) | 1,3 |

С помощью программы автоматизации расчета метрик Холстеда были подсчитаны операторы и операнды в программе на С, определены измеримые характеристики программы на С, определены расчетные характеристики программы на С. Результаты представлены в таблице 10, 11, 12.

Таблица 10. Программный расчет операторов и операндов в программе на C.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Оператор | Количество | № | Операнд | Количество |
| 1 | != | 1 | 1 | 0 | 4 |
| 2 | && | 1 | 2 | 1 | 3 |
| 3 | () | 15 | 3 | 1.0E-6 | 1 |
| 4 | \* | 11 | 4 | 14 | 1 |
| 5 | + | 9 | 5 | 15 | 1 |
| 6 | ++ | 1 | 6 | 16 | 1 |
| 7 | , | 6 | 7 | 2 | 7 |
| 8 | - | 5 | 8 | 3 | 1 |
| 9 | / | 8 | 9 | 4 | 1 |
| 10 | ; | 30 | 10 | 7 | 1 |
| 11 | <= | 2 | 11 | 9 | 1 |
| 12 | = | 20 | 12 | delta\_x | 7 |
| 13 | \_& | 1 | 13 | end\_cor | 2 |
| 14 | \_\* | 6 | 14 | end\_sum | 3 |
| 15 | \_- | 3 | 15 | even\_sum | 4 |
| 16 | \_\_\* | 1 | 16 | i | 4 |
| 17 | dfx | 3 | 17 | lower | 9 |
| 18 | dowhile | 1 | 18 | odd\_sum | 7 |
| 19 | exp | 2 | 19 | pieces | 6 |
| 20 | fabs | 2 | 20 | sum | 9 |
| 21 | for | 1 | 21 | sum1 | 4 |
| 22 | fx | 5 | 22 | tol | 4 |
| 23 | 1 | main | 23 | upper | 7 |
| 24 | return | 3 | 24 | x | 7 |
| 25 | simps | 2 |  |  |  |

Таблица 11. Программное определение измеримых характеристик программы на C.

|  |  |  |
| --- | --- | --- |
| Характеристика | Формула | Значение |
| Число уникальных операторов | η1 | 25 |
| Число уникальных операндов | η2 | 24 |
| Число всех операторов | N1 | 140 |
| Число всех операндов | N2 | 95 |
| Словарь программы | η1 = η1 + η2 | 49 |
| Длина программы | N = N1 + N2 | 235 |

Таблица 12. Программное определение расчетных характеристик программы на C.

|  |  |  |
| --- | --- | --- |
| Характеристика | Формула | Значение |
| Теоретическая оценка длины программы | Ň=η1log2η1+η2log2η2 | 226,136 |
| Реальный объём | 𝑉 = 𝑁 log2η | 1319,46 |
| Потенциальный объём | V\*=(2+η2\*)log2(2+η2\*) | 15,51 |
| Уровень программы | *L=V\*/V* | 0,012 |
| Интеллектуальное содержание программы | *I=2η2/(η1N2)(N1+N2)log2(η1+η2)* | 26,7 |
| Работа программиста | *E=V/L* | 112250 |
| Время программирования | *T=E/S* | 11225 |
| Уровень языка | 𝜆 =*LV\** | 0.18 |
| Ожидаемое число ошибок | B=(*V\**)2/(1000𝜆) | 0.8 |

C помощью команды:

gcc -S C.c -masm=intel -fno-asynchronous-unwind-tables

Получен ассемблерный код этой программы. В полученном коде удалены комментарии и отладочные директивы. Ассемблерный код представлен в приложении Г. Для ассемблерной программы был произведен ручной расчет операторов и операндов. Результат представлен в таблице 13.

Таблица 13. Ручной расчет операторов и операндов в программе на ассемблере.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Оператор | Количество | № | Операнд | Количество |
| 1 | endbr64 | 4 | 1 | rbp | 8 |
| 2 | push | 4 | 2 | rsp | 8 |
| 3 | mov | 28 | 3 | 16 | 2 |
| 4 | sub | 5 | 4 | QWORD PTR -8[rbp] | 4 |
| 5 | movsd | 53 | 5 | xmm0 | 91 |
| 6 | movq | 12 | 6 | xmm1 | 38 |
| 7 | xorpd | 3 | 7 | QWORD PTR .LC0[rip] | 3 |
| 8 | divsd | 7 | 8 | QWORD PTR .LC1[rip] | 3 |
| 9 | call exp@PLT | 2 | 9 | 112 | 1 |
| 10 | leave | 4 | 10 | QWORD PTR -72[rbp] | 6 |
| 11 | ret | 4 | 11 | QWORD PTR -80[rbp] | 4 |
| 12 | cvtsi2sd | 3 | 12 | QWORD PTR -88[rbp] | 1 |
| 13 | addsd | 9 | 13 | QWORD PTR -96[rbp] | 6 |
| 14 | call fx | 4 | 14 | DWORD PTR -64[rbp] | 4 |
| 15 | pxor | 3 | 15 | xmm2 | 8 |
| 16 | call dfx | 2 | 16 | rdi | 2 |
| 17 | movapd | 3 | 17 | 2 | 1 |
| 18 | mulsd | 9 | 18 | QWORD PTR -40[rbp] | 5 |
| 19 | subsd | 4 | 19 | rax | 24 |
| 20 | sal | 1 | 20 | QWORD PTR -56[rbp] | 4 |
| 21 | jmp .L6 | 1 | 21 | QWORD PTR -48[rbp] | 2 |
| 22 | add | 3 | 22 | QWORD PTR -104[rbp] | 2 |
| 23 | shr | 1 | 23 | QWORD PTR -32[rbp] | 2 |
| 24 | sar | 1 | 24 | xmm3 | 3 |
| 25 | cmp | 1 | 25 | QWORD PTR -24[rbp] | 2 |
| 26 | jle .L7 | 1 | 26 | QWORD PTR .LC3[rip] | 1 |
| 27 | ucomisd | 2 | 27 | QWORD PTR .LC4[rip] | 1 |
| 28 | jp .L11 | 1 | 28 | QWORD PTR [rax] | 4 |
| 29 | je .L8 | 1 | 29 | QWORD PTR -16[rbp] | 4 |
| 30 | andpd | 2 | 30 | DWORD PTR -60[rbp] | 1 |
| 31 | comisd | 1 | 31 | 1 | 3 |
| 32 | jnb .L10 | 1 | 32 | eax | 13 |
| 33 | nop | 1 | 33 | QWORD PTR -8[rbp] | 4 |
| 34 | xor | 2 | 34 | edx | 3 |
| 35 | call simps | 1 | 35 | 31 | 1 |
| 36 | je .L14 | 1 | 36 | QWORD PTR .LC5[rip] | 1 |
|  |  |  | 37 | QWORD PTR .LC6[rip] | 1 |
|  |  |  | 38 | QWORD PTR .LC7[rip] | 1 |
|  |  |  | 39 | QWORD PTR .LC8[rip] | 1 |
|  |  |  | 40 | QWORD PTR .LC9[rip] | 1 |
|  |  |  | 41 | 32 | 1 |
|  |  |  | 42 | QWORD PTR fs:40 | 2 |
|  |  |  | 43 | QWORD PTR .LC10[rip] | 1 |
|  |  |  | 44 | QWORD PTR .LC11[rip] | 1 |
|  |  |  | 45 | QWORD PTR .LC12[rip] | 1 |
|  |  |  | 46 | -32[rbp] | 3 |
|  |  |  | 47 | rdx | 2 |
|  |  |  | 48 | 0 | 1 |
|  |  |  | 49 | rcx | 2 |

Далее был произведен расчет измеримых характеристик этой программы. Полученный результат представлен в таблице 14.

Таблица 14. Ручное определение измеримых характеристик программы на ассемблере.

|  |  |  |
| --- | --- | --- |
| Характеристика | Формула | Значение |
| Число уникальных операторов | η1 | 36 |
| Число уникальных операндов | η2 | 49 |
| Число всех операторов | N1 | 185 |
| Число всех операндов | N2 | 288 |
| Словарь программы | η1 = η1 + η2 | 85 |
| Длина программы | N = N1 + N2 | 473 |

Далее были произведены расчеты для получения расчетных характеристик. Результаты представлены в таблице 15. Для расчетов значение коэффициента Стауда S принято 10.

Таблица 15. Ручное определение расчетных характеристик программы на ассемблере.

|  |  |  |
| --- | --- | --- |
| Характеристика | Формула | Значение |
| Теоретическая оценка длины программы | Ň=η1log2η1+η2log2η2 | 461,24 |
| Реальный объём | 𝑉 = 𝑁 log2η | 3031,64 |
| Потенциальный объём | V\*=(2+η2\*)log2(2+η2\*) | 15,51 |
| Уровень программы | *L=V\*/V* | 0,005 |
| Интеллектуальное содержание программы | *I=2η2/(η1N2)(N1+N2)log2(η1+η2)* | 28,66 |
| Работа программиста | *E=V/L* | 606328 |
| Время программирования | *T=E/S* | 60632,8 |
| Уровень языка | 𝜆 =*LV\** | 0,078 |
| Ожидаемое число ошибок | B=(*V\**)2/(1000𝜆) | 3,08 |

В таблице 16 приведена сводная характеристика расчётов для трёх языков (Паскаль, Си, Ассемблер):

Таблица 16 - Сводная таблица расчётов по трём языкам

|  |  |  |  |
| --- | --- | --- | --- |
| **Характеристика** | **Паскаль** | **Си** | **Ассемблер** |
| Число уникальных операторов | 22 | 25 | 36 |
| Число уникальных операндов | 26 | 24 | 49 |
| Число всех операторов | 134 | 142 | 185 |
| Число всех операндов | 83 | 84 | 288 |
| Словарь программы | 48 | 49 | 85 |
| Длина программы | 217 | 226 | 473 |
| Теоретическая оценка длины программы | 220,32 | 226,14 | 461,24 |
| Реальный объём | 1020 | 1268,92 | 3031,64 |
| Потенциальный объём | 15,51 | 15,51 | 15,51 |
| Уровень программы | 0,015 | 0,012 | 0,005 |
| Интеллектуальное содержание программы | 34,51 | 29 | 28,66 |
| Работа программиста | 68000 | 105743,3 | 606328 |
| Время программирования | 6800 | 10574,33 | 60632,8 |
| Уровень языка | 0,233 | 0,186 | 0,078 |
| Ожидаемое число ошибок | 1,03 | 1,3 | 3,08 |

**Выводы.**

В ходе выполнения лабораторной работы были изучены метрические характеристики качества разработки программ на основе метрик Холстеда. В результате были вручную рассчитаны метрики Холстеда для программ на Pascal, C и ассемблере, аналогичные расчёты были произведены с помощью специальных программ автоматизации расчёта для языков Pascal и C. На основе полученных характеристик было установлено, что программа на ассемблере обладает гораздо большим объёмом и, следовательно, требует гораздо больше времени для написания, что также увеличивает число потенциальных ошибок в ней.

**Приложение А.**

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;

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

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

sum := (end\_sum+4\*odd\_sum)\*delta\_x / 3;

repeat

pieces := pieces\*2;

sum1 := sum;

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

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

odd\_sum := 0;

for i := 1 to pieces div 2 do

begin

x := lower+delta\_x\*(2\*i-1);

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

end;

sum := (7\*end\_sum+14\*even\_sum+16\*odd\_sum

+end\_cor\*delta\_x)\*delta\_x / 15;

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

end; { simps }

begin { main program }

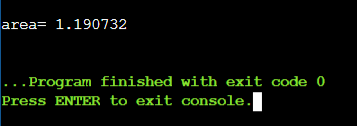
lower := 1;

upper := 9;

simps(lower,upper,tol,sum);

end.

**Приложение Б.**



**Приложение В.**

#include <stdio.h>

#include <math.h>

const double tol = 1.0E-6;

double fx(double x) {

return exp(-x / 2);

}

double dfx(double x) {

return -(exp(-x / 2)) / 2;

}

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

int pieces=2;

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

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

double even\_sum =0;

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

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

\*sum=(end\_sum+4\*odd\_sum)\*delta\_x / 3;

double sum1;

double x;

do

{

pieces=pieces\*2;

sum1=\*sum;

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

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

odd\_sum=0;

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

x=lower+delta\_x\*(2\*i-1);

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

}

\*sum=(7\*end\_sum+14\*even\_sum+16\*odd\_sum+end\_cor\*delta\_x)\*delta\_x / 15;

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

}

int main()

{

double lower=1;

double upper=9;

double sum = 0;

simps(lower,upper,tol,&sum);

return 0;

}

**Приложение Г.**

fx:

endbr64

push rbp

mov rbp, rsp

sub rsp, 16

movsd QWORD PTR -8[rbp], xmm0

movsd xmm0, QWORD PTR -8[rbp]

movq xmm1, QWORD PTR .LC0[rip]

xorpd xmm0, xmm1

movsd xmm1, QWORD PTR .LC1[rip]

divsd xmm0, xmm1

call exp@PLT

leave

ret

.size fx, .-fx

.globl dfx

.type dfx, @function

dfx:

endbr64

push rbp

mov rbp, rsp

sub rsp, 16

movsd QWORD PTR -8[rbp], xmm0

movsd xmm0, QWORD PTR -8[rbp]

movq xmm1, QWORD PTR .LC0[rip]

xorpd xmm0, xmm1

movsd xmm1, QWORD PTR .LC1[rip]

divsd xmm0, xmm1

call exp@PLT

movq xmm1, QWORD PTR .LC0[rip]

xorpd xmm0, xmm1

movsd xmm1, QWORD PTR .LC1[rip]

divsd xmm0, xmm1

leave

ret

.size dfx, .-dfx

.globl simps

.type simps, @function

simps:

endbr64

push rbp

mov rbp, rsp

sub rsp, 112

movsd QWORD PTR -72[rbp], xmm0

movsd QWORD PTR -80[rbp], xmm1

movsd QWORD PTR -88[rbp], xmm2

mov QWORD PTR -96[rbp], rdi

mov DWORD PTR -64[rbp], 2

movsd xmm0, QWORD PTR -80[rbp]

subsd xmm0, QWORD PTR -72[rbp]

cvtsi2sd xmm1, DWORD PTR -64[rbp]

divsd xmm0, xmm1

movsd QWORD PTR -40[rbp], xmm0

movsd xmm0, QWORD PTR -72[rbp]

addsd xmm0, QWORD PTR -40[rbp]

call fx

movq rax, xmm0

mov QWORD PTR -56[rbp], rax

pxor xmm0, xmm0

movsd QWORD PTR -48[rbp], xmm0

mov rax, QWORD PTR -72[rbp]

movq xmm0, rax

call fx

movsd QWORD PTR -104[rbp], xmm0

mov rax, QWORD PTR -80[rbp]

movq xmm0, rax

call fx

addsd xmm0, QWORD PTR -104[rbp]

movsd QWORD PTR -32[rbp], xmm0

mov rax, QWORD PTR -72[rbp]

movq xmm0, rax

call dfx

movsd QWORD PTR -104[rbp], xmm0

mov rax, QWORD PTR -80[rbp]

movq xmm0, rax

call dfx

movsd xmm3, QWORD PTR -104[rbp]

subsd xmm3, xmm0

movapd xmm0, xmm3

movsd QWORD PTR -24[rbp], xmm0

movsd xmm1, QWORD PTR -56[rbp]

movsd xmm0, QWORD PTR .LC3[rip]

mulsd xmm0, xmm1

addsd xmm0, QWORD PTR -32[rbp]

mulsd xmm0, QWORD PTR -40[rbp]

movsd xmm1, QWORD PTR .LC4[rip]

divsd xmm0, xmm1

mov rax, QWORD PTR -96[rbp]

movsd QWORD PTR [rax], xmm0

.L10:

sal DWORD PTR -64[rbp]

mov rax, QWORD PTR -96[rbp]

movsd xmm0, QWORD PTR [rax]

movsd QWORD PTR -16[rbp], xmm0

movsd xmm0, QWORD PTR -80[rbp]

subsd xmm0, QWORD PTR -72[rbp]

cvtsi2sd xmm1, DWORD PTR -64[rbp]

divsd xmm0, xmm1

movsd QWORD PTR -40[rbp], xmm0

movsd xmm0, QWORD PTR -48[rbp]

addsd xmm0, QWORD PTR -56[rbp]

movsd QWORD PTR -48[rbp], xmm0

pxor xmm0, xmm0

movsd QWORD PTR -56[rbp], xmm0

mov DWORD PTR -60[rbp], 1

jmp .L6

.L7:

mov eax, DWORD PTR -60[rbp]

add eax, eax

sub eax, 1

cvtsi2sd xmm0, eax

mulsd xmm0, QWORD PTR -40[rbp]

movsd xmm1, QWORD PTR -72[rbp]

addsd xmm0, xmm1

movsd QWORD PTR -8[rbp], xmm0

mov rax, QWORD PTR -8[rbp]

movq xmm0, rax

call fx

movsd xmm1, QWORD PTR -56[rbp]

addsd xmm0, xmm1

movsd QWORD PTR -56[rbp], xmm0

add DWORD PTR -60[rbp], 1

.L6:

mov eax, DWORD PTR -64[rbp]

mov edx, eax

shr edx, 31

add eax, edx

sar eax

cmp DWORD PTR -60[rbp], eax

jle .L7

movsd xmm1, QWORD PTR -32[rbp]

movsd xmm0, QWORD PTR .LC5[rip]

mulsd xmm1, xmm0

movsd xmm2, QWORD PTR -48[rbp]

movsd xmm0, QWORD PTR .LC6[rip]

mulsd xmm0, xmm2

addsd xmm1, xmm0

movsd xmm2, QWORD PTR -56[rbp]

movsd xmm0, QWORD PTR .LC7[rip]

mulsd xmm0, xmm2

addsd xmm1, xmm0

movsd xmm0, QWORD PTR -24[rbp]

mulsd xmm0, QWORD PTR -40[rbp]

addsd xmm0, xmm1

mulsd xmm0, QWORD PTR -40[rbp]

movsd xmm1, QWORD PTR .LC8[rip]

divsd xmm0, xmm1

mov rax, QWORD PTR -96[rbp]

movsd QWORD PTR [rax], xmm0

mov rax, QWORD PTR -96[rbp]

movsd xmm0, QWORD PTR [rax]

ucomisd xmm0, QWORD PTR -16[rbp]

jp .L11

ucomisd xmm0, QWORD PTR -16[rbp]

je .L8

.L11:

mov rax, QWORD PTR -96[rbp]

movsd xmm0, QWORD PTR [rax]

subsd xmm0, QWORD PTR -16[rbp]

movq xmm1, QWORD PTR .LC9[rip]

andpd xmm1, xmm0

mov rax, QWORD PTR -96[rbp]

movsd xmm0, QWORD PTR [rax]

mulsd xmm0, QWORD PTR -88[rbp]

movq xmm2, QWORD PTR .LC9[rip]

andpd xmm0, xmm2

comisd xmm0, xmm1

jnb .L10

.L8:

nop

leave

ret

.size simps, .-simps

.globl main

.type main, @function

main:

endbr64

push rbp

mov rbp, rsp

sub rsp, 32

mov rax, QWORD PTR fs:40

mov QWORD PTR -8[rbp], rax

xor eax, eax

movsd xmm0, QWORD PTR .LC10[rip]

movsd QWORD PTR -24[rbp], xmm0

movsd xmm0, QWORD PTR .LC11[rip]

movsd QWORD PTR -16[rbp], xmm0

pxor xmm0, xmm0

movsd QWORD PTR -32[rbp], xmm0

movsd xmm1, QWORD PTR .LC12[rip]

lea rdx, -32[rbp]

movsd xmm0, QWORD PTR -16[rbp]

mov rax, QWORD PTR -24[rbp]

mov rdi, rdx

movapd xmm2, xmm1

movapd xmm1, xmm0

movq xmm0, rax

call simps

mov eax, 0

mov rcx, QWORD PTR -8[rbp]

xor rcx, QWORD PTR fs:40

je .L14

call \_\_stack\_chk\_fail@PLT

.L14:

leave

ret

.size main, .-main

.section .rodata

.align 16