МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по практической работе №1
по дисциплине «Качество и метрология программного обеспечения»
Тема: Расчет метрических характеристик качества разработки программ по метрикам Холстеда

Студент гр. 7304 _____ Соколов И.Д.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Расчет метрических характеристик качества разработки программ по метрикам Холстеда для программ на языках Pascal, C, Assembler.

Постановка задачи.

Для заданного варианта процедуры, реализованной на языке Pascal, разработать аналогичный алгоритм и реализовать его на языках программирования C и Assembler (в ассемблерном представлении програмы нужно удалить директивы описаний и отладочные директивы, оставив только исполняемые операторы).

Для каждой из программ 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 вхождения j-го оператора в тексте программы;
- число вхождения j-го операнда в тексте программы;
- словарь программы;
- длина 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

Вариант задания – 18 Вычисление функции ошибок распределения Гаусса

Программа на языке Pascal:

```
{ Программа 18. Вычисление функции ошибок распределения Гаусса(вар.2). }

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 }
  done := false;
  x := 2;
  repeat
    if x < 0.0 then
      done := true
    else
    begin
      if x = 0.0 then
      begin
```

```
      er := 0.0;
      ec := 1.0;
    end
    else 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 };
    x := x - 1;
  end { if }
 until done;
end.
```

Выполнение работы.

Код программы на Паскаль был переписан на язык Си:

```
float x, er, ec;
unsigned char done;

float erf(float x)
 /* infinite series expansion of the Gaussian error function */

{
 static const float sqrtpi = 1.7724538;
 static const float tol = 1.0E-4;

 float x2, sum, sum1, term;
 int i;


 float erf_result;
 x2 = x * x;
```

```
  sum = x;
  term = x;
  i = 0;
  do {
    i = i + 1;
    sum1 = sum;
    term = 2.0 * term * x2 / (1.0 + 2.0 * i);
    sum = term + sum1;
  } while (!(term < tol * sum));
  erf_result = 2.0 * sum * exp(-x2) / sqrtpi;
  return erf_result;
}     /* erf */

float erfc(float x)
  /* complement of error function */
{
  static const float sqrtpi = 1.7724538;
  int terms = 12;

  float x2, u, v, sum;
  int i;

  float erfc_result;
  x2 = x * x;
  v = 1.0 / (2.0 * x2);
  u = 1.0 + v * (terms + 1.0);
  for( i = terms; i >= 1; i --)
  {
    sum = 1.0 + i * v / u;
    u = sum;
  }
  erfc_result = exp(-x2) / (x * sum * sqrtpi);
  return erfc_result;
}       /* ercf */

int main()
{       /* main */
```

```
 done = 0;
 x = 2;
 do {
  if (x < 0.0)
    done = 1;
  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 */
    x = x - 1;
  }   /* if */
 } while (!done);
 return 0;
}
```

Версия программы на языке Assember x86-64 gcc10.2:

```
erf(float):
        push    rbp
        mov     rbp, rsp
        sub     rsp, 48
        movss   DWORD PTR [rbp-36], xmm0
        movss   xmm0, DWORD PTR [rbp-36]
        mulss   xmm0, xmm0
        movss   DWORD PTR [rbp-16], xmm0
```

```
        movss   xmm0, DWORD PTR [rbp-36]
        movss   DWORD PTR [rbp-4], xmm0
        movss   xmm0, DWORD PTR [rbp-36]
        movss   DWORD PTR [rbp-8], xmm0
        mov     DWORD PTR [rbp-12], 0
.L6:
        add     DWORD PTR [rbp-12], 1
        movss   xmm0, DWORD PTR [rbp-4]
        movss   DWORD PTR [rbp-20], xmm0
        pxor    xmm0, xmm0
        cvtss2sd        xmm0, DWORD PTR [rbp-8]
        movapd  xmm1, xmm0
        addsd   xmm1, xmm0
        pxor    xmm0, xmm0
        cvtss2sd        xmm0, DWORD PTR [rbp-16]
        mulsd   xmm1, xmm0
        pxor    xmm0, xmm0
        cvtsi2sd        xmm0, DWORD PTR [rbp-12]
        movapd  xmm2, xmm0
        addsd   xmm2, xmm0
        movsd   xmm0, QWORD PTR .LC0[rip]
        addsd   xmm0, xmm2
        divsd   xmm1, xmm0
        pxor    xmm0, xmm0
        cvtsd2ss        xmm0, xmm1
        movss   DWORD PTR [rbp-8], xmm0
        movss   xmm0, DWORD PTR [rbp-8]
        addss   xmm0, DWORD PTR [rbp-20]
        movss   DWORD PTR [rbp-4], xmm0
        movss   xmm1, DWORD PTR [rbp-4]
        movss   xmm0, DWORD PTR .LC1[rip]
        mulss   xmm1, xmm0
        movss   xmm0, DWORD PTR [rbp-8]
        comiss  xmm0, xmm1
        jb      .L9
        jmp     .L6
.L9:
```

```
        pxor    xmm0, xmm0
        cvtss2sd        xmm0, DWORD PTR [rbp-4]
        addsd   xmm0, xmm0
        movsd   QWORD PTR [rbp-48], xmm0
        movss   xmm0, DWORD PTR [rbp-16]
        movss   xmm1, DWORD PTR .LC2[rip]
        movaps  xmm4, xmm0
        xorps   xmm4, xmm1
        movd    eax, xmm4
        movd    xmm0, eax
        call    std::exp(float)
        cvtss2sd        xmm0, xmm0
        mulsd   xmm0, QWORD PTR [rbp-48]
        movsd   xmm1, QWORD PTR .LC3[rip]
        divsd   xmm0, xmm1
        cvtsd2ss        xmm0, xmm0
        movss   DWORD PTR [rbp-24], xmm0
        movss   xmm0, DWORD PTR [rbp-24]
        leave
        ret
erfc(float):
        push    rbp
        mov     rbp, rsp
        sub     rsp, 48
        movss   DWORD PTR [rbp-36], xmm0
        mov     DWORD PTR [rbp-16], 12
        movss   xmm0, DWORD PTR [rbp-36]
        mulss   xmm0, xmm0
        movss   DWORD PTR [rbp-20], xmm0
        pxor    xmm0, xmm0
        cvtss2sd        xmm0, DWORD PTR [rbp-20]
        movapd  xmm1, xmm0
        addsd   xmm1, xmm0
        movsd   xmm0, QWORD PTR .LC0[rip]
        divsd   xmm0, xmm1
        cvtsd2ss        xmm0, xmm0
        movss   DWORD PTR [rbp-24], xmm0
```

```
        pxor    xmm1, xmm1
        cvtss2sd        xmm1, DWORD PTR [rbp-24]
        pxor    xmm2, xmm2
        cvtsi2sd        xmm2, DWORD PTR [rbp-16]
        movsd   xmm0, QWORD PTR .LC0[rip]
        addsd   xmm0, xmm2
        mulsd   xmm1, xmm0
        movsd   xmm0, QWORD PTR .LC0[rip]
        addsd   xmm0, xmm1
        cvtsd2ss        xmm0, xmm0
        movss   DWORD PTR [rbp-4], xmm0
        mov     eax, DWORD PTR [rbp-16]
        mov     DWORD PTR [rbp-12], eax
.L12:
        cmp     DWORD PTR [rbp-12], 0
        jle     .L11
        pxor    xmm0, xmm0
        cvtsi2ss        xmm0, DWORD PTR [rbp-12]
        mulss   xmm0, DWORD PTR [rbp-24]
        movaps  xmm1, xmm0
        divss   xmm1, DWORD PTR [rbp-4]
        movss   xmm0, DWORD PTR .LC4[rip]
        addss   xmm0, xmm1
        movss   DWORD PTR [rbp-8], xmm0
        movss   xmm0, DWORD PTR [rbp-8]
        movss   DWORD PTR [rbp-4], xmm0
        sub     DWORD PTR [rbp-12], 1
        jmp     .L12
.L11:
        movss   xmm0, DWORD PTR [rbp-20]
        movss   xmm1, DWORD PTR .LC2[rip]
        xorps   xmm0, xmm1
        movd    eax, xmm0
        movd    xmm0, eax
        call    std::exp(float)
        movd    eax, xmm0
        movss   xmm0, DWORD PTR [rbp-36]
```

```
        movaps  xmm1, xmm0
        mulss   xmm1, DWORD PTR [rbp-8]
        movss   xmm0, DWORD PTR .LC5[rip]
        mulss   xmm1, xmm0
        movd    xmm0, eax
        divss   xmm0, xmm1
        movss   DWORD PTR [rbp-28], xmm0
        movss   xmm0, DWORD PTR [rbp-28]
        leave
        ret
main:
        push    rbp
        mov     rbp, rsp
        mov     BYTE PTR done[rip], 0
        movss   xmm0, DWORD PTR .LC6[rip]
        movss   DWORD PTR x[rip], xmm0
.L24:
        movss   xmm1, DWORD PTR x[rip]
        pxor    xmm0, xmm0
        comiss  xmm0, xmm1
        jbe     .L29
        mov     BYTE PTR done[rip], 1
        jmp     .L17
.L29:
        movss   xmm0, DWORD PTR x[rip]
        pxor    xmm1, xmm1
        ucomiss xmm0, xmm1
        jp      .L18
        pxor    xmm1, xmm1
        ucomiss xmm0, xmm1
        jne     .L18
        pxor    xmm0, xmm0
        movss   DWORD PTR er[rip], xmm0
        movss   xmm0, DWORD PTR .LC4[rip]
        movss   DWORD PTR ec[rip], xmm0
        jmp     .L20
.L18:
```

```
        movss   xmm1, DWORD PTR x[rip]
        movss   xmm0, DWORD PTR .LC8[rip]
        comiss  xmm0, xmm1
        jbe     .L30
        mov     eax, DWORD PTR x[rip]
        movd    xmm0, eax
        call    erf(float)
        movd    eax, xmm0
        mov     DWORD PTR er[rip], eax
        movss   xmm1, DWORD PTR er[rip]
        movss   xmm0, DWORD PTR .LC4[rip]
        subss   xmm0, xmm1
        movss   DWORD PTR ec[rip], xmm0
        jmp     .L20
.L30:
        mov     eax, DWORD PTR x[rip]
        movd    xmm0, eax
        call    erfc(float)
        movd    eax, xmm0
        mov     DWORD PTR ec[rip], eax
        movss   xmm1, DWORD PTR ec[rip]
        movss   xmm0, DWORD PTR .LC4[rip]
        subss   xmm0, xmm1
        movss   DWORD PTR er[rip], xmm0
.L20:
        movss   xmm0, DWORD PTR x[rip]
        movss   xmm1, DWORD PTR .LC4[rip]
        subss   xmm0, xmm1
        movss   DWORD PTR x[rip], xmm0
.L17:
        movzx   eax, BYTE PTR done[rip]
        test    al, al
        jne     .L23
        jmp     .L24
.L23:
        mov     eax, 0
        pop     rbp
```

ret

Были определены метрические характеристики программы на языке Pascal:

а) Ручной расчет

Таблица 1 - измеримые характеристики Pascal

Операторы			Операнды		
№	Число вхождений	Оператор	№	Число вхождений	Операнд
1	8	()	1	2	1.7724538
2	6	+	2	1	1.0E-4
3	5	-	3	1	erf
4	5	/	4	3	12
5	28	;	5	1	0
6	3	<	6	3	0.0
7	26	:=	7	2	1
8	13	*	8	8	1.0
9	1	=	9	5	2.0
10	2	repeat..until	10	1	1.5
10	1	for	11	2	8
11	1	erf	12	1	4
12	1	erfc	13	18	x
13	8	begin...end	14	6	er
14	3	if…then...else	15	5	ec
15	2	exp	16	4	done

			17	5	sqrtpi
			18	2	tol
			19	8	x2
			20	10	sum
			21	3	sum1
			22	6	term
			23	7	i
			24	3	terms
			25	4	u
			26	4	v
			27	1	false
			29	1	true
			30	1	erfc
Итого	113		Итого:	118	

η_1 | Количество уникальных операторов = 15

η_2 | Количество уникальных операндов = 30

η | Словарь программы = 15 + 30 = 45

N | Длина программы = 113 + 118 = 231

Расчетные характеристики:

Оценка длины программы:

$\grave{N} = \eta_1 \log_2\eta_1 + \eta_2 \log_2\eta_2 = 15\log_2(15) + 30\log_2(30) = 205{,}810$

Объем программы:

Реальный $V = N \log_2 \eta = 231\log_2(45) = 1268{,}618$

Потенциальный $V^* = (2+\eta_2^*)\log_2(2+\eta_2^*) = 11{,}609$ ($\eta_2^* = 3$, так как у программы erfd3 3 внешних параметра: x, erf, erfc)

Уровень программы:

$L = V^* / V = 11{,}609 / 1268{,}618 = 0{,}0091$

Оценка уровня программы:

$\acute{L} = 2 * \eta_2 / (\eta_1 * N_2) = 2 * 30 / (15 * 118) = 0{,}0338$

Интеллектуальное содержание программы:

$I = \acute{L} * V = 0{,}0338 * 1268{,}618 = 42{,}879$

Работа программиста:

$E = V^2/V^* = 1268{,}618^{\,2} / \ 11{,}609 = 138633{,}0975$

Время программирования:

$T = E/S = 138633{,}0975/10 = 13863{,}30975$

Уровень используемого языка программирования:

$\lambda = L \times V^* = 0{,}0091 * 11{,}609 = 0{,}1056$

Ожидаемое число ошибок в программе:

$B = V / 1000 = 1268{,}618/ 1000 = 1{,}26$

Таблица 2 - расчетные характеристики вручную Pascal

Длина программы	231
Реальный объем программы	1268,618
Уровень программы	0,0091
Оценка уровня программы	0,0338
Интеллектуальное содержание программы	42,879
Работа программиста	138633,0975
Время программирования	13863,30975

Уровень используемого языка программирования	0,1056
Ожидаемое число ошибок в программе	2

б) Автоматический расчет

Таблица автоматического расчета

==========

The number of different operators : 16
The number of different operands : 29
The total number of operators : 95
The total number of operands : 113

Dictionary (D) : 45
Length (N) : 208
Length estimation (^N) : 204.881
Volume (V) : 1142.31
Potential volume (*V) : 11.6096
Limit volume (**V) : 15.6844
Programming level (L) : 0.0101633
Programming level estimation (^L) : 0.0320796
Intellect (I) : 36.6448
Time of programming (T) : 6244.15
Time estimation (^T) : 1948.59
Programming language level (lambda) : 0.117993
Work on programming (E) : 112395
Error (B) : 0.776323
Error estimation (^B) : 0.380768

Table:

==========

Operators:

| 1 | 16 | ()

| 2 | 13 | *
| 3 | 6 | +
| 4 | 5 | -
| 5 | 5 | /
| 6 | 3 | <
| 7 | 30 | =
| 8 | 2 | const
| 9 | 2 | erf
| 10 | 2 | erfc
| 11 | 2 | exp
| 12 | 1 | for
| 13 | 3 | if
| 14 | 1 | program
| 15 | 2 | real
| 16 | 2 | repeat

Operands:

| 1 | 1 | 0
| 2 | 3 | 0.0
| 3 | 3 | 1
| 4 | 8 | 1.0
| 5 | 1 | 1.0E-4
| 6 | 1 | 1.5
| 7 | 2 | 1.7724538
| 8 | 1 | 12
| 9 | 1 | 2
| 10 | 4 | 2.0
| 11 | 4 | done
| 12 | 5 | ec
| 13 | 5 | er
| 14 | 1 | erf
| 15 | 1 | erfc
| 16 | 1 | erfd3
| 17 | 1 | false
| 18 | 7 | i
| 19 | 4 | sqrtpi
| 20 | 10 | sum
| 21 | 3 | sum1

| 22 | 6 | term
| 23 | 3 | terms
| 24 | 2 | tol
| 25 | 1 | true
| 26 | 4 | u
| 27 | 4 | v
| 28 | 18 | x
| 29 | 8 | x2

Summary:

==========

The number of different operators : 16
The number of different operands : 29
The total number of operators : 95
The total number of operands : 113

Dictionary (D) : 45
Length (N) : 208
Length estimation (^N) : 204.881
Volume (V) : 1142.31
Potential volume (*V) : 11.6096
Limit volume (**V) : 15.6844
Programming level (L) : 0.0101633
Programming level estimation (^L) : 0.0320796
Intellect (I) : 36.6448
Time of programming (T) : 6244.15
Time estimation (^T) : 1948.59
Programming language level (lambda) : 0.117993
Work on programming (E) : 112395
Error (B) : 0.776323
Error estimation (^B) : 0.380768

Были определены метрические характеристики программы на языке С:

Таблица 3 - измеримые характеристики C

Операторы			Операнды		
№	Число вхождений	Оператор	№	Число вхождений	Операнд
1	5	-	1	16	x
2	14	()	2	5	er
3	30	=	3	5	ec
4	8	{}	4	4	done
5	2	<	5	3	0
6	36	;	6	3	0.0
7	1	!	7	4	1
8	1	erf	8	8	1.0
9	1	erfc	9	1	1.5
10	3	if..else	10	4	2.0
11	2	do..while	11	2	1.7724538
12	1	==	12	1	1.0E-4
13	3	return	13	8	x2
14	2	>=	14	10	sum
15	13	*	15	3	sum1
16	6	+	16	6	term
17	5	/	17	10	i
18	2	exp	18	3	erf_result

19	1	--	19	4	sqrtpi
20	1	for ..	20	2	tol
			21	2	erf
			22	2	erfc
			23	3	terms
			24	4	u
			25	4	v
			26	3	erfc_result
			27	1	12
Итого	137		Итого	121	

η_1 | Количество уникальных операторов = 20

η_2 | Количество уникальных операндов = 27

η | Словарь программы = 20 + 27 = 47

N | Длина программы = 137 + 121 = 258

Расчетные характеристики:

Оценка длины программы:

$\grave{N} = \eta_1 \log_2\eta_1 + \eta_2 \log_2\eta_2 = 20\log_2(20) + 27\log_2(27) = 214{,}820$

Объем программы:

Реальный $V = N \log_2 \eta = 258\log_2(47) = 1433{,}083$

Потенциальный $V^* = (2+\eta_2^*)\log_2(2+\eta_2^*) = 11{,}609$ ($\eta_2^* = 3$, так как у программы erfd3 3 внешних параметра: x, erf, erfc)

Уровень программы:

$L = V^* / V = 11{,}609 / 1433{,}083 = 0{,}0081$

Оценка уровня программы:

$\acute{L} = 2 * \eta_2 / (\eta_1 * N_2) = 2 * 27 / (20 * 121) = 0,0223$

Интеллектуальное содержание программы:

$I = \acute{L} * V = 0,0223 * 1433,083 = 31,9577$

Работа программиста:

$E = V^2/V^* = 1433,083^{\,2} / \ 11,609 = 176908,164$

Время программирования:

$T = E/S = 176908,164/10 = 17690,8164$

Уровень используемого языка программирования:

$\lambda = L \times V^* = 0,0081 * 11,609 = 0,0940$

Ожидаемое число ошибок в программе:

$B = V / 1000 = 1433,083/ 1000 = 1,433$

Таблица 4 - расчетные характеристики вручную С

Длина программы	258
Реальный объем программы	1433,083
Уровень программы	0,0081
Оценка уровня программы	0,0223
Интеллектуальное содержание программы	31,9577
Работа программиста	176908,164
Время программирования	17690,8164
Уровень используемого языка программирования	0,0940
Ожидаемое число ошибок в программе	2

б) Автоматический расчет

Таблица автоматического расчета:

==========

The number of different operators : 21

The number of different operands : 26

The total number of operators : 104

The total number of operands : 120

Dictionary (D) : 47

Length (N) : 224

Length estimation (^N) : 214.45

Volume (V) : 1244.23

Potential volume (*V) : 11.6096

Limit volume (**V) : 15.6844

Programming level (L) : 0.0093308

Programming level estimation (^L) : 0.0206349

Intellect (I) : 25.6745

Time of programming (T) : 7408.13

Time estimation (^T) : 3207.03

Programming language level (lambda) : 0.108327

Work on programming (E) : 133346

Error (B) : 0.870027

Error estimation (^B) : 0.414743

Table:

==========

Operators:

| 1 | 1 | !

| 2 | 14 | ()

| 3 | 13 | *

| 4 | 6 | +
| 5 | 8 | ,
| 6 | 3 | -
| 7 | 1 | --
| 8 | 5 | /
| 9 | 2 | <
| 10 | 30 | =
| 11 | 1 | ==
| 12 | 2 | >=
| 13 | 2 | _-
| 14 | 2 | dowhile
| 15 | 2 | erf
| 16 | 2 | erfc
| 17 | 2 | exp
| 18 | 1 | for
| 19 | 3 | if
| 20 | 1 | main
| 21 | 3 | return

Operands:

| 1 | 3 | 0
| 2 | 3 | 0.0
| 3 | 4 | 1
| 4 | 8 | 1.0
| 5 | 1 | 1.0E-4
| 6 | 1 | 1.5
| 7 | 2 | 1.7724538
| 8 | 1 | 12
| 9 | 1 | 2
| 10 | 4 | 2.0

| 11 | 4 | done
| 12 | 5 | ec
| 13 | 5 | er
| 14 | 3 | erf_result
| 15 | 3 | erfc_result
| 16 | 10 | i
| 17 | 4 | sqrtpi
| 18 | 10 | sum
| 19 | 3 | sum1
| 20 | 6 | term
| 21 | 3 | terms
| 22 | 2 | tol
| 23 | 4 | u
| 24 | 4 | v
| 25 | 18 | x
| 26 | 8 | x2

Summary:

==========

The number of different operators : 21

The number of different operands : 26

The total number of operators : 104

The total number of operands : 120

Dictionary (D) : 47

Length (N) : 224

Length estimation (^N) : 214.45

Volume (V) : 1244.23

Potential volume (*V) : 11.6096

Limit volume (**V) : 15.6844

Programming level (L) : 0.0093308

Programming level estimation (^L) : 0.0206349

Intellect (I) : 25.6745

Time of programming (T) : 7408.13

Time estimation (^T) : 3207.03

Programming language level (lambda) : 0.108327

Work on programming (E) : 133346

Error (B) : 0.870027

Error estimation (^B) : 0.414743

Были определены метрические характеристики программы на языке Assembler:

Таблица 5 - измеримые характеристики Assembler

Операторы			Операнды		
№	Число вхождений	Оператор	№	Число вхождений	Операнд
1	3	push	1	7	rbp
2	14	mov	2	5	rsp
3	3	sub	3	4	48
4	52	movss	4	7	DWORD PTR [rbp-36]
5	13	pxor	5	130	xmm0
6	3	comiss	6	6	DWORD PTR [rbp-16]
7	2	ucomiss	7	8	DWORD PTR [rbp-4]

8	10	movd	8	8	DWORD PTR [rbp-8]
9	3	subss	9	7	DWORD PTR [rbp-12]
10	1	cmp	10	4	0
11	2	leave	11	3	1
12	3	ret	12	5	DWORD PTR [rbp-20]
13	1	jne .L14	13	45	xmm1
14	1	jmp .L15	14	7	xmm2
15	6	mulss	15	4	QWORD PTR .LC0[rip]
16	1	add	16	1	DWORD PTR .LC1[rip]
17	6	cvtss2sd	17	2	QWORD PTR [rbp-48]
18	3	movapd	18	2	DWORD PTR .LC2[rip]
19	7	addsd	19	3	xmm4
20	3	mulsd	20	18	eax
21	2	cvtsi2sd	21	1	QWORD PTR .LC3[rip]
22	3	divsd	22	5	DWORD PTR [rbp-24]
23	4	cvtsd2ss	23	1	12

24	1	jb .L9	24	5	DWORD PTR .LC4[rip]
25	1	jmp .L6	25	1	DWORD PTR .LC5[rip]
26	3	movaps	26	2	DWORD PTR [rbp-28]
27	2	xorps	27	3	BYTE PTR done[rip]
28	2	call std::exp(float)	28	1	DWORD PTR .LC6[rip]
29	1	cmp	29	8	DWORD PTR x[rip]
30	1	jle .L11	30	4	DWORD PTR er[rip]
31	1	cvtsi2ss	31	4	DWORD PTR ec[rip]
32	2	divss	32	1	DWORD PTR .LC8[rip]
33	2	addss	33	2	al
34	6	mulss			
35	1	jmp .L12			
36	1	jbe .L30			
37	2	jmp .L20			
38	1	movzx			
39	1	test			
40	1	jne .L23			

41	1	jmp .L24			
42	1	pop			
43	1	call erf(float)			
44	1	call erfc(float)			
Итого	179		Итого:	314	

η_1 | Количество уникальных операторов = 44

η_2 | Количество уникальных операндов = 33

η | Словарь программы = 44 + 33 = 77

N | Длина программы = 179 + 314 = 493

Расчетные характеристики:

Оценка длины программы:

$\grave{N} = \eta_1 \log_2\eta_1 + \eta_2 \log_2\eta_2 = 44\log_2(44) + 33\log_2(33) = 406{,}679$

Объем программы:

Реальный $V = N \log_2 \eta = 493\log_2(77) = 3089{,}525$

Потенциальный $V^* = (2+\eta_2^*)\log_2(2+\eta_2^*) = 11{,}609$ ($\eta_2^* = 3$, так как у программы erfd3 3 внешних параметра: x, erf, erfc)

Уровень программы:

$L = V^* / V = 11{,}609 / 3089{,}525 = 0{,}00375$

Оценка уровня программы:

$\acute{L} = 2 * \eta_2 / (\eta_1 * N_2) = 2 * 33 / (44 * 314) = 0{,}00477$

Интеллектуальное содержание программы:

$I = \acute{L} * V = 0{,}00477 * 3089{,}525 = 14{,}7370$

Работа программиста:

$E = V^2/V^* = 3089{,}525^{\,2} / 11{,}609 = 822221{,}097$

Время программирования:

T = E/S = 822221,097/10 = 82222,1097

Уровень используемого языка программирования:

$\lambda = L \times V^* = 0{,}00375 * 11{,}609 = 0{,}04353$

Ожидаемое число ошибок в программе:

B= V / 1000 = 3089,525/ 1000 = 3,089

Таблица 6 - расчетные характеристики вручную Assembler

Длина программы	493
Реальный объем программы	3089,525
Уровень программы	0,00375
Оценка уровня программы	0,00477
Интеллектуальное содержание программы	14,7370
Работа программиста	822221,097
Время программирования	82222,1097
Уровень используемого языка программирования	0,04353
Ожидаемое число ошибок в программе	4

Сводные таблицы по программам на Pascal, C, Assembler:

Таблица 3 - измеримые характеристики сводная

	Pascal	C	ASM
Количество уникальных операторов	15	20	44
Количество уникальных операндов	30	27	33
Общее количество операторов	113	137	179
Общее количество операндов	118	121	314

Словарь программы	45	47	77
Длина программы	231	258	493

Таблица 4 - расчетные характеристики сводная вручную

	Pascal	C	ASM
Длина программы	231	258	493
Реальный объем программы	1268,618	1433,083	3089,525
Уровень программы	0,0091	0,0081	0,00375
Оценка уровня программы	0,0338	0,0223	0,00477
Интеллектуальное содержание программы	42,879	31,9577	14,7370
Работа программиста	138633,0975	176908,164	822221,097
Время программирования	13863,30975	17690,8164	82222,1097
Уровень используемого языка программирования	0,1056	0,0940	0,04353
Ожидаемое число ошибок в программе	2	2	4

Таблица 5 - расчетные характеристики сводная автоматически

	Pascal	C
Количество уникальных операторов	16	21
Количество уникальных операндов	29	26
Общее количество операторов	95	104
Общее количество операндов	113	120

Длина программы	208	224
Словарь программы	45	47
Оценка длины программы	204.881	214.45
Реальный объем программы	1142.31	1244.23
Уровень программы	0.0101633	0.0093308
Оценка уровня программы	0.0320796	0.0206349
Интеллектуальное содержание программы	36.6448	25.6745
Работа программиста	112395	133346
Время программирования	6244.15	7408.13
Уровень используемого языка программирования	0.117993	0.108327
Ожидаемое число ошибок в программе	0.380768	0.414743

Выводы.

В результате выполнения работы был получен код на языках программирования C и Assembler, предлагаемого к рассмотрению алгоритма на языке Pascal. Для всех программ были оценены метрические характеристики (измеримые и расчетные) по Холстеду. Для характеристик были также рассчитаны их оценки (по заданию).

Из метрических характеристик можно сделать вывод о том, что реализация на языке C или Pascal требует от программиста меньших усилий, а также меньшего времени на программирование по сравнению с реализациями на

Assembler. При этом реализация на Pascal обладает наименьшей реальной длиной программы и наименьшим ожидаемым числом ошибок. Но если учесть, что стандартная библиотека C включает в себя функции erf и erfc, то для реализации данного алгоритма более предпочтительным является язык C, так как использование функций из стандартной библиотеки повлияет в положительную сторону для всех метрик, включая длину программы, работу программиста и ожидаемое количество ошибок.