**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**

Студентка гр. 7304 _____ Каляева А.В.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

**Цель работы:**

Расчет и сравнение метрик Холстеда для программ, написанных на языках Паскаль, Си, Ассемблер.

**Задание:**

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.

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.

## Ход работы:

### 1. Расчет метрик вручную

Программа на языке Паскаль, а также реализованные программы на языках Си и Ассемблер представлены в приложениях А, Б и В соответственно.

В таблицах 1-3 представлены результаты подсчета количества операторов и операндов для программ, написанных на языках Паскаль, Си, Ассемблер.

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

| № | Оператор | Число вхождений | № | Операнд | Число вхождений |
|---|---|---|---|---|---|
| **1** | ; | 15 | **1** | 80 | 1 |
| **2** | begin…end | 6 | **2** | max | 2 |
| **3** | := | 12 | **3** | x | 2 |
| **4** | for... to… do | 2 | **4** | i | 6 |
| **5** | if… then | 1 | **5** | j | 4 |
| **6** | repeat… until | 1 | **6** | n | 6 |
| **7** | + | 1 | **7** | hold | 2 |
| **8** | while…end | 1 | **8** | a | 4 |
| **9** | > | 2 | **9** | jump | 5 |
| **10** | [] | 5 | **10** | p | 2 |
| **11** | div | 1 | **11** | q | 2 |
| **12** | swap | 1 | **12** | 1 | 4 |
| **13** | sort | 1 | **13** | 2 | 1 |
| **14** | random | 1 | **14** | false | 1 |
| **15** | randomize | 1 | **15** | true | 1 |
| **16** | () | 5 | **16** | done | 3 |
| **17** | >= | 1 | **17** | 100 | 1 |

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

| № | Оператор | Число вхождений | № | Операнд | Число вхождений |
|---|---|---|---|---|---|
| **1** | sort | 1 | **1** | 100 | 1 |
| **2** | swap | 1 | **2** | 80 | 1 |
| **3** | for | 2 | **3** | i | 6 |
| **4** | if…else | 1 | **4** | j | 6 |
| **5** | [] | 5 | **5** | hold | 2 |
| **6** | = | 13 | **6** | p | 2 |
| **7** | / | 1 | **7** | q | 2 |
| **8** | + | 1 | **8** | done | 3 |

| | | | | | |
|---|---|---|---|---|---|
| **9** | < | 2 | **9** | a | 5 |
| **10** | > | 3 | **10** | n | 6 |
| **11** | ; | 19 | **11** | jump | 5 |
| **12** | return | 1 | **12** | max | 3 |
| **13** | * | 4 | **13** | true | 1 |
| **14** | rand | 1 | **14** | false | 1 |
| **15** | ++ | 2 | | | |
| **16** | & | 2 | | | |
| **17** | () | 11 | | | |
| **18** | && | 1 | | | |
| **19** | while | 2 | | | |
| **20** | ! | 1 | | | |

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

| № | Оператор | Число вхождений | № | Операнд | Число вхождений |
|---|---|---|---|---|---|
| **1** | push | 4 | **1** | rbp | 4 |
| **2** | mov | 64 | **2** | rsp | 9 |
| **3** | movss | 9 | **3** | QWORD PTR [rbp-24] | 9 |
| **4** | nop | 1 | **4** | rdi | 5 |
| **5** | pop | 1 | **5** | QWORD PTR [rbp-32] | 3 |
| **6** | ret | 3 | **6** | rsi | 3 |
| **7** | sub | 7 | **7** | rax | 29 |
| **8** | jmp | 3 | **8** | xmm0 | 12 |
| **9** | shr | 3 | **9** | DWORD PTR [rax] | 6 |
| **10** | add | 10 | **10** | DWORD PTR [rbp-4] | 2 |
| **11** | sar | 3 | **11** | 32 | 2 |
| **12** | cmp | 4 | **12** | DWORD PTR [rbp-28] | 7 |
| **13** | jle | 1 | **13** | esi | 2 |
| **14** | cdqe | 5 | **14** | eax | 34 |
| **15** | lea | 5 | **15** | DWORD PTR [rbp-8] | 5 |
| **16** | comiss | 1 | **16** | .L3 | 2 |
| **17** | jbe | 1 | **17** | edx | 15 |
| **18** | call swap | 1 | **18** | 31 | 2 |
| **19** | jl | 2 | **19** | BYTE PTR [rbp-1] | 3 |
| **20** | movzx | 1 | **20** | 1 | 6 |
| **21** | xor | 1 | **21** | DWORD PTR [rbp-12] | 6 |
| **22** | test | 1 | **22** | .L4 | 2 |
| **23** | jne | 1 | **23** | 0 | 8 |
| **24** | jg | 1 | **24** | DWORD PTR [rbp-16] | 4 |

| | | | | | |
|---|---|---|---|---|---|
| **25** | leave | 2 | **25** | .L5 | 3 |
| **26** | movsx | 5 | **26** | [0+rax*4] | 5 |
| **27** | div | 1 | **27** | rdx | 21 |
| **28** | imul | 3 | **28** | rcx | 2 |
| **29** | sal | 1 | **29** | .L7 | 2 |
| **30** | call rand | 1 | **30** | al | 2 |
| **31** | pxor | 1 | **31** | .L8 | 2 |
| **32** | cvtsi2ss | 1 | **32** | .L9 | 2 |
| **33** | call sort | 1 | **33** | rbx | 4 |
| | | | **34** | 40 | 1 |
| | | | **35** | 80 | 1 |
| | | | **36** | QWORD PTR [rbp-40] | 1 |
| | | | **37** | r8 | 1 |
| | | | **38** | r9d | 1 |
| | | | **39** | 16 | 3 |
| | | | **40** | 3 | 1 |
| | | | **41** | 2 | 2 |
| | | | **42** | QWORD PTR [rbp-48] | 3 |
| | | | **43** | DWORD PTR [rbp-20] | 4 |
| | | | **44** | .L13 | 2 |
| | | | **45** | 1374389535 | 1 |
| | | | **46** | 5 | 1 |
| | | | **47** | 100 | 1 |
| | | | **48** | ecx | 5 |
| | | | **49** | DWORD PTR [rax+rdx*4] | 1 |
| | | | **50** | .L14 | 2 |

В таблице 4 представлены результаты расчета метрик Холстеда вручную для программ, реализованных на языках Паскаль, Си, Ассемблер.

Таблица 4 – Результаты расчета метрик вручную.

| **Характеристики** | **Паскаль** | **Си** | **Ассемблер** |
|---|---|---|---|
| Число уникальных операторов | 17 | 20 | 33 |
| Число уникальных операндов | 17 | 14 | 50 |
| Общее число операторов | 57 | 74 | 149 |
| Общее число операндов | 47 | 44 | 254 |
| Алфавит | 34 | 34 | 83 |
| Экспериментальная длина программы | 104 | 118 | 403 |
| Теоретическая длина программы | 138,974 | 139,738 | 448,662 |
| Объем программы | 529,048 | 600,266 | 2569,125 |
| Потенциальный объем | 11,6 | 11,6 | 11,6 |

| | | | |
|---|---|---|---|
| Уровень программы | 0,022 | 0,019 | 0,005 |
| Интеллектуальное содержание | 22,515 | 19,101 | 30,651 |
| Ожидание уровня программы | 0,043 | 0,032 | 0,012 |
| Работа по программированию | 24047,636 | 31041,860 | 568534,825 |
| Ожидание времени кодирования | 2404,764 | 3104,186 | 56853,483 |
| Уровень языка программирования | 0,255 | 0,220 | 0,058 |
| Уровень ошибок | 0,5 | 0,6 | 2,3 |

## 2. Программный расчет метрик

Результаты программного расчета метрик для программ, реализованных на языках Паскаль, Си представлены в приложениях Г и Д соответственно.

В таблицах 5-6 представлены результаты программного подсчета количества операторов и операндов для программ, написанных на языках Паскаль, Си.

Таблица 5 – Количество операторов и операндов в программе, написанной на языке Паскаль.

| № | Оператор | Число вхождений | № | Операнд | Число вхождений |
|---|---|---|---|---|---|
| **1** | () | 8 | **1** | 1 | 4 |
| **2** | + | 1 | **2** | 100 | 1 |
| **3** | / | 1 | **3** | 2 | 1 |
| **4** | ; | 30 | **4** | 80 | 1 |
| **5** | = | 12 | **5** | Shel_sort | 1 |
| **6** | > | 2 | **6** | a | 5 |
| **7** | [] | 5 | **7** | ary | 1 |
| **8** | const | 1 | **8** | done | 4 |
| **9** | for | 2 | **9** | false | 1 |
| **10** | if | 1 | **10** | hold | 3 |
| **11** | program | 1 | **11** | i | 6 |
| **12** | random | 1 | **12** | j | 4 |
| **13** | randomize | 1 | **13** | jump | 6 |
| **14** | repeat | 1 | **14** | max | 3 |
| **15** | sort | 2 | **15** | n | 7 |
| **16** | swap | 2 | **16** | p | 3 |
| **17** | type | 1 | **17** | q | 3 |
| **18** | while | 1 | **18** | true | 1 |
| | | | **19** | x | 3 |

Таблица 6 – Количество операторов и операндов в программе, написанной на языке Си.

| № | Оператор | Число вхождений | № | Операнд | Число вхождений |
|---|---|---|---|---|---|
| **1** | ! | 1 | **1** | 0 | 4 |
| **2** | % | 1 | **2** | 1 | 1 |
| **3** | && | 1 | **3** | 100 | 1 |
| **4** | () | 12 | **4** | 2 | 1 |
| **5** | + | 1 | **5** | 80 | 1 |
| **6** | ++ | 2 | **6** | a | 6 |
| **7** | , | 4 | **7** | done | 4 |
| **8** | / | 1 | **8** | hold | 2 |
| **9** | ; | 26 | **9** | i | 9 |
| **10** | < | 2 | **10** | j | 6 |
| **11** | = | 13 | **11** | jump | 5 |
| **12** | > | 3 | **12** | max | 3 |
| **13** | [] | 5 | **13** | n | 7 |
| **14** | _& | 2 | **14** | p | 3 |
| **15** | _* | 4 | **15** | q | 3 |
| **16** | _[] | 1 | **16** | x | 3 |
| **17** | __* | 4 | **17** | | |
| **18** | do…while | 1 | **18** | | |
| **19** | for | 2 | **19** | | |
| **20** | if | 1 | | | |
| **21** | main | 1 | | | |
| **22** | rand | 1 | | | |
| **23** | return | 1 | | | |
| **24** | sort | 2 | | | |
| **25** | swap | 2 | | | |
| **26** | while | 1 | | | |

В таблице 7 представлены результаты программного расчета метрик Холстеда для программ, реализованных на языках Паскаль, Си.

Таблица 7 – Результаты программного расчета метрик.

| Характеристики | Паскаль | Си |
|---|---|---|
| Число уникальных операторов | 18 | 26 |
| Число уникальных операндов | 19 | 16 |
| Общее число операторов | 73 | 95 |
| Общее число операндов | 58 | 59 |
| Алфавит | 37 | 42 |
| Экспериментальная длина программы | 131 | 154 |

| | | |
|---|---|---|
| Теоретическая длина программы | 155,769 | 186,211 |
| Объем программы | 682,438 | 830,417 |
| Потенциальный объем | 11,6096 | 11,6096 |
| Уровень программы | 0,017012 | 0,0139805 |
| Интеллектуальное содержание | 24,8397 | 17,3229 |
| Ожидание уровня программы | 0,0363985 | 0,0208605 |
| Работа по программированию | 40115,1 | 59398,2 |
| Ожидание времени кодирования | 1238,56 | 2674,14 |
| Уровень языка программирования | 0,197503 | 0,162309 |
| Уровень ошибок | 0,390617 | 0,507451 |

## 3. Сравнение полученных результатов

В таблице 8 представлены результаты программного и ручного расчета метрик Холстеда для программ, реализованных на языках Паскаль, Си.

Таблица 8 – Сводная таблица расчетов на языках Паскаль, Си.

| **Характеристики** | **Ручной расчет Паскаль** | **Программный расчет Паскаль** | **Ручной расчет Си** | **Программный расчет Си** |
|---|---|---|---|---|
| Число уникальных операторов | 17 | 18 | 20 | 26 |
| Число уникальных операндов | 17 | 19 | 14 | 16 |
| Общее число операторов | 57 | 73 | 74 | 95 |
| Общее число операндов | 47 | 58 | 44 | 59 |
| Алфавит | 34 | 37 | 34 | 42 |
| Экспериментальная длина программы | 104 | 131 | 118 | 154 |
| Теоретическая длина программы | 138,974 | 155,769 | 139,738 | 186,211 |
| Объем программы | 529,048 | 682,438 | 600,266 | 830,417 |
| Потенциальный объем | 11,6 | 11,6096 | 11,6 | 11,6096 |
| Уровень программы | 0,022 | 0,017012 | 0,019 | 0,0139805 |
| Интеллектуальное содержание | 22,515 | 24,8397 | 19,101 | 17,3229 |
| Ожидание уровня программы | 0,043 | 0,0363985 | 0,032 | 0,0208605 |
| Работа по программированию | 24047,636 | 40115,1 | 31041,860 | 59398,2 |

| | | | | |
|---|---|---|---|---|
| Ожидание времени кодирования | 2404,764 | 1238,56 | 3104,186 | 2674,14 |
| Уровень языка программирования | 0,255 | 0,197503 | 0,220 | 0,162309 |
| Уровень ошибок | 0,5 | 0,390617 | 0,6 | 0,507451 |

**Выводы:**

Метрические характеристики программ, написанных на языках Си и Паскаль выглядят похожим образом, так они имеют схожую структуру. Характеристики программы на языке Ассемблер сильно отличаются. Это связано с тем, что язык Ассемблер является языком низкого уровня.

В ходе выполнения данной работы все характеристики были посчитаны вручную и автоматически. Различия в полученных результатах обусловлены тем, что автоматический метод считает не только функциональную часть программы, но и объявления типов переменных и функций. Также различия для программы на языке Си обусловлены тем, что инструмент автоматического подсчета не имеет имеет возможности обработки типа данных bool, который присутствует в коде программы.

## ПРИЛОЖЕНИЕ А.

## КОД ПРОГРАММЫ НА ЯЗЫКЕ ПАСКАЛЬ.

```
program Shell_sort;
const   max     = 80;

type    ary     = array[1..max]of real;

var     x           : ary;
        i,n         : integer;

procedure sort(var a: ary; n: integer);

var   done   : boolean;
      jump,i,j: integer;

procedure swap(var p,q: real);
var   hold   : real;

begin
  hold:=p;
  p:=q;
  q:=hold
end;

begin
  jump:=n;
  while jump>1 do
    begin
      jump:=jump div 2;
      repeat
       done:=true;
       for j:=1 to n do
         begin
           i:=j+jump;
           if (a[j]>a[i]) then
             begin
              swap(a[j],a[i]);
              done:=false
           end
       end
      until done
  end
end;

begin
 n:=max;
 randomize;
 for i:=1 to n do
   x[i]:= random(100);
 sort( x,n );
end.
```

## ПРИЛОЖЕНИЕ Б.

## КОД ПРОГРАММЫ НА ЯЗЫКЕ СИ.

```
void swap(float* p, float* q) {
    float hold = *p;
    *p = *q;
    *q = hold;
}

float* sort(float* a, int n) {
    int i;
    int done;
    int jump = n;
    while (jump > 0) {
        jump = jump / 2;
        do {
            done = 1;
        for (int j = 0; j < n; j++) {
            i = j + jump;
            if ((n>i) && (a[j] > a[i])) {
                swap(&a[i], &a[j]);
                done = 0;
            }
        }
        } while(!done);
    }

    return a;
}

int main()
{
    int max = 80;

    int n = max;
    float x[max];

    for (int i = 0; i < n; ++i) {
        x[i] = (float) (rand() % 100);
    }

    sort(x, n);
}
```

## ПРИЛОЖЕНИЕ В.

## КОД ПРОГРАММЫ НА ЯЗЫКЕ АССЕМБЛЕР.

```
swap:
        push    rbp
        mov     rbp, rsp
        mov     QWORD PTR [rbp-24], rdi
        mov     QWORD PTR [rbp-32], rsi
        mov     rax, QWORD PTR [rbp-24]
        movss   xmm0, DWORD PTR [rax]
        movss   DWORD PTR [rbp-4], xmm0
        mov     rax, QWORD PTR [rbp-32]
        movss   xmm0, DWORD PTR [rax]
        mov     rax, QWORD PTR [rbp-24]
        movss   DWORD PTR [rax], xmm0
        mov     rax, QWORD PTR [rbp-32]
        movss   xmm0, DWORD PTR [rbp-4]
        movss   DWORD PTR [rax], xmm0
        nop
        pop     rbp
        ret
sort:
        push    rbp
        mov     rbp, rsp
        sub     rsp, 32
        mov     QWORD PTR [rbp-24], rdi
        mov     DWORD PTR [rbp-28], esi
        mov     eax, DWORD PTR [rbp-28]
        mov     DWORD PTR [rbp-8], eax
        jmp     .L3
.L9:
        mov     eax, DWORD PTR [rbp-8]
        mov     edx, eax
        shr     edx, 31
        add     eax, edx
        sar     eax
        mov     DWORD PTR [rbp-8], eax
.L8:
        mov     BYTE PTR [rbp-1], 1
        mov     DWORD PTR [rbp-12], 0
        jmp     .L4
.L7:
        mov     edx, DWORD PTR [rbp-12]
        mov     eax, DWORD PTR [rbp-8]
        add     eax, edx
        mov     DWORD PTR [rbp-16], eax
        mov     eax, DWORD PTR [rbp-28]
        cmp     eax, DWORD PTR [rbp-16]
        jle     .L5
        mov     eax, DWORD PTR [rbp-12]
        cdqe
        lea     rdx, [0+rax*4]
```

```
        mov     rax, QWORD PTR [rbp-24]
        add     rax, rdx
        movss   xmm0, DWORD PTR [rax]
        mov     eax, DWORD PTR [rbp-16]
        cdqe
        lea     rdx, [0+rax*4]
        mov     rax, QWORD PTR [rbp-24]
        add     rax, rdx
        movss   xmm1, DWORD PTR [rax]
        comiss  xmm0, xmm1
        jbe     .L5
        mov     eax, DWORD PTR [rbp-12]
        cdqe
        lea     rdx, [0+rax*4]
        mov     rax, QWORD PTR [rbp-24]
        add     rdx, rax
        mov     eax, DWORD PTR [rbp-16]
        cdqe
        lea     rcx, [0+rax*4]
        mov     rax, QWORD PTR [rbp-24]
        add     rax, rcx
        mov     rsi, rdx
        mov     rdi, rax
        call    swap
        mov     BYTE PTR [rbp-1], 0
.L5:
        add     DWORD PTR [rbp-12], 1
.L4:
        mov     eax, DWORD PTR [rbp-12]
        cmp     eax, DWORD PTR [rbp-28]
        jl      .L7
        movzx   eax, BYTE PTR [rbp-1]
        xor     eax, 1
        test    al, al
        jne     .L8
.L3:
        cmp     DWORD PTR [rbp-8], 0
        jg      .L9
        mov     rax, QWORD PTR [rbp-24]
        leave
        ret
main:
        push    rbp
        mov     rbp, rsp
        push    rbx
        sub     rsp, 40
        mov     rax, rsp
        mov     rbx, rax
        mov     DWORD PTR [rbp-24], 80
        mov     eax, DWORD PTR [rbp-24]
        mov     DWORD PTR [rbp-28], eax
        mov     eax, DWORD PTR [rbp-24]
        movsx   rdx, eax
```

```
        sub     rdx, 1
        mov     QWORD PTR [rbp-40], rdx
        movsx   rdx, eax
        mov     r8, rdx
        mov     r9d, 0
        movsx   rdx, eax
        mov     rsi, rdx
        mov     edi, 0
        cdqe
        lea     rdx, [0+rax*4]
        mov     eax, 16
        sub     rax, 1
        add     rax, rdx
        mov     edi, 16
        mov     edx, 0
        div     rdi
        imul    rax, rax, 16
        sub     rsp, rax
        mov     rax, rsp
        add     rax, 3
        shr     rax, 2
        sal     rax, 2
        mov     QWORD PTR [rbp-48], rax
        mov     DWORD PTR [rbp-20], 0
        jmp     .L13
.L14:
        call    rand
        movsx   rdx, eax
        imul    rdx, rdx, 1374389535
        shr     rdx, 32
        sar     edx, 5
        mov     ecx, eax
        sar     ecx, 31
        sub     edx, ecx
        imul    ecx, edx, 100
        sub     eax, ecx
        mov     edx, eax
        pxor    xmm0, xmm0
        cvtsi2ss        xmm0, edx
        mov     rax, QWORD PTR [rbp-48]
        mov     edx, DWORD PTR [rbp-20]
        movsx   rdx, edx
        movss   DWORD PTR [rax+rdx*4], xmm0
        add     DWORD PTR [rbp-20], 1
.L13:
        mov     eax, DWORD PTR [rbp-20]
        cmp     eax, DWORD PTR [rbp-28]
        jl      .L14
        mov     edx, DWORD PTR [rbp-28]
        mov     rax, QWORD PTR [rbp-48]
        mov     esi, edx
        mov     rdi, rax
        call    sort
```

```
        mov     rsp, rbx
        mov     eax, 0
        mov     rbx, QWORD PTR [rbp-8]
        leave
ret
```

## ПРИЛОЖЕНИЕ Г.

## РЕЗУЛЬТАТ ПРОГРАММНОГО РАСЧЕТА МЕТРИК ДЛЯ ПРОГРАММЫ НА ЯЗЫКЕ ПАСКАЛЬ.

```
Statistics for module ./output.lxm
=====================================
The number of different operators     : 18
The number of different operands      : 19
The total number of operators         : 73
The total number of operands          : 58

Dictionary                    (  D)   : 37
Length                        (  N)   : 131
Length estimation             ( ^N)   : 155.769
Volume                        (  V)   : 682.438
Potential volume              ( *V)   : 11.6096
Limit volume                  (**V)   : 15.6844
Programming level             (  L)   : 0.017012
Programming level estimation  ( ^L)   : 0.0363985
Intellect                     (  I)   : 24.8397
Time of programming           (  T)   : 2228.62
Time estimation               ( ^T)   : 1238.56
Programming language level    (lambda) : 0.197503
Work on programming           (  E)   : 40115.1
Error                         (  B)   : 0.390617
Error estimation              ( ^B)   : 0.227479


Table:
====================================
Operators:
|  1    |  8    | ()
|  2    |  1    | +
|  3    |  1    | /
|  4    |  30   | ;
|  5    |  12   | =
|  6    |  2    | >
|  7    |  5    | []
|  8    |  1    | const
|  9    |  2    | for
|  10   |  1    | if
|  11   |  1    | program
|  12   |  1    | random
|  13   |  1    | randomize
|  14   |  1    | repeat
|  15   |  2    | sort
|  16   |  2    | swap
|  17   |  1    | type
|  18   |  1    | while
Operands:
|  1    |  4    | 1
```

```
|  2    |  1    | 100
|  3    |  1    | 2
|  4    |  1    | 80
|  5    |  1    | Shell_sort
|  6    |  5    | a
|  7    |  1    | ary
|  8    |  4    | done
|  9    |  1    | false
|  10   |  3    | hold
|  11   |  6    | i
|  12   |  4    | j
|  13   |  6    | jump
|  14   |  3    | max
|  15   |  7    | n
|  16   |  3    | p
|  17   |  3    | q
|  18   |  1    | true
|  19   |  3    | x
```

```
Summary:
=====================================
The number of different operators     : 18
The number of different operands      : 19
The total number of operators         : 73
The total number of operands          : 58

Dictionary                    (  D)    : 37
Length                        (  N)    : 131
Length estimation             ( ^N)    : 155.769
Volume                        (  V)    : 682.438
Potential volume              ( *V)    : 11.6096
Limit volume                  (**V)    : 15.6844
Programming level             (  L)    : 0.017012
Programming level estimation  ( ^L)    : 0.0363985
Intellect                     (  I)    : 24.8397
Time of programming           (  T)    : 2228.62
Time estimation               ( ^T)    : 1238.56
Programming language level    (lambda) : 0.197503
Work on programming           (  E)    : 40115.1
Error                         (  B)    : 0.390617
Error estimation              ( ^B)    : 0.227479
```

## ПРИЛОЖЕНИЕ Д.

## РЕЗУЛЬТАТ ПРОГРАММНОГО РАСЧЕТА МЕТРИК ДЛЯ ПРОГРАММЫ НА ЯЗЫКЕ СИ.

```
Statistics for module ./output.lxm
=====================================
The number of different operators     : 26
The number of different operands      : 16
The total number of operators         : 95
The total number of operands          : 59

Dictionary                    (  D)   : 42
Length                        (  N)   : 154
Length estimation             ( ^N)   : 186.211
Volume                        (  V)   : 830.417
Potential volume              ( *V)   : 11.6096
Limit volume                  (**V)   : 15.6844
Programming level             (  L)   : 0.0139805
Programming level estimation  ( ^L)   : 0.0208605
Intellect                     (  I)   : 17.3229
Time of programming           (  T)   : 3299.9
Time estimation               ( ^T)   : 2674.14
Programming language level    (lambda) : 0.162309
Work on programming           (  E)   : 59398.2
Error                         (  B)   : 0.507451
Error estimation              ( ^B)   : 0.276806


Table:
=====================================
Operators:
|  1    |  1    | !
|  2    |  1    | %
|  3    |  1    | &&
|  4    |  12   | ()
|  5    |  1    | +
|  6    |  2    | ++
|  7    |  4    | ,
|  8    |  1    | /
|  9    |  26   | ;
|  10   |  2    | <
|  11   |  13   | =
|  12   |  3    | >
|  13   |  5    | []
|  14   |  2    | _&
|  15   |  4    | _*
|  16   |  1    | _[]
|  17   |  4    | __*
|  18   |  1    | dowhile
|  19   |  2    | for
|  20   |  1    | if
```

```
|   21    |   1     | main
|   22    |   1     | rand
|   23    |   1     | return
|   24    |   2     | sort
|   25    |   2     | swap
|   26    |   1     | while
Operands:
|   1     |   4     | 0
|   2     |   1     | 1
|   3     |   1     | 100
|   4     |   1     | 2
|   5     |   1     | 80
|   6     |   6     | a
|   7     |   4     | done
|   8     |   2     | hold
|   9     |   9     | i
|   10    |   6     | j
|   11    |   5     | jump
|   12    |   3     | max
|   13    |   7     | n
|   14    |   3     | p
|   15    |   3     | q
|   16    |   3     | x


Summary:
=====================================
The number of different operators     : 26
The number of different operands      : 16
The total number of operators         : 95
The total number of operands          : 59

Dictionary                    (  D)    : 42
Length                        (  N)    : 154
Length estimation             ( ^N)    : 186.211
Volume                        (  V)    : 830.417
Potential volume              ( *V)    : 11.6096
Limit volume                  (**V)    : 15.6844
Programming level             (  L)    : 0.0139805
Programming level estimation  ( ^L)    : 0.0208605
Intellect                     (  I)    : 17.3229
Time of programming           (  T)    : 3299.9
Time estimation               ( ^T)    : 2674.14
Programming language level    (lambda) : 0.162309
Work on programming           (  E)    : 59398.2
Error                         (  B)    : 0.507451
Error estimation              ( ^B)    : 0.276806
```