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: Расчёт метрических характеристик качества разработки программ по метрикам Холстеда

Студент гр. 7304 _____ Есиков О.И.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

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

Ход выполнения.

Был выбран 5 вариант задания – процедура сортировки методом «пузырька». Из исходного варианта программы был удалён вызов процедуры печати массива – с целью оптимизации для последующих работ. Исходный код полученной программы представлен в Приложении А. Для этой программы был произведён вручную расчёт операторов и операндов. Результат представлен в Таблице 1.

№	Оператор	Количество	№	Операнд	Количество
1	;	15	*1*	max	3
2	:=	10	*2*	80	1
3	=	2	*3*	ary	1
4	>	1	*4*	1	5
5	() или begin end	12	*5*	hold	2
6	[]	7	*6*	p	2
7	if … then	1	*7*	q	2
8	for … to … do	3	*8*	no_change	3
9	procedure sort	1	*9*	true	1
10	procedure swap	1	*10*	j	3
11	procedure write_arr	1	*11*	n	5
12	+	2	*12*	a	2
13	-	1	*13*	false	1
14	writeln	2	*14*	i *(в Main)*	2
15	write(x[i]:7:1, ' ')	1	*15*	x	2
16	randomize	1	*16*	i *(в процедуре write_arr)*	1
17	random(100)	1			
18	.	1			
19	sort(x, n)	1			
20	swap(a[j], a[j+1])	1			
21	repeat … until	1			

Таблица 1 – Ручной расчёт операторов и операндов в программе на Pascal

Затем был произведён расчёт измеримых характеристик этой программы. Полученный результат представлен в Таблице 2.

Характеристика	Формула	Значение
Число уникальных операторов	η_1	21
Число уникальных операндов	η_2	16
Число всех операторов	N_1	66
Число всех операндов	N_2	36
Словарь программы	$\eta = \eta_1 + \eta_2$	37
Длина программы	$N = N_1 + N_2$	102

Таблица 2 – Ручное определение измеримых характеристик программы на Pascal

После чего были получены расчётные характеристики программы на Pascal. Результат представлен в Таблице 3.

Для расчётов значение коэффициента Стауда *S* принято 10; значение η_2^* принято 2, поскольку исследуемая процедура *sort* принимает 2 аргумента и не имеет возвращаемого значения.

Характеристика	Формула	Значение
Теоретическая оценка длины программы	$\check{N} = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2$	156,24
Реальный объём	$V = N \log_2 \eta$	531,36
Потенциальный объём	$V^* = (2 + \eta_2^*) \log_2 (2 + \eta_2^*)$	8
Уровень программы	$L = \frac{V^*}{V}$	0,015
Интеллектуальное содержание программы	$I = \frac{2\eta_2}{\eta_1 N_2} (N_1 + N_2) \log_2 (\eta_1 + \eta_2)$	22,49
Работа в программировании	$E = \frac{V}{L}$	35293
Время программирования	$T = \frac{E}{S}$	3529
Уровень языка	$\lambda = LV^*$	0,120
Ожидаемое число ошибок	$B = \frac{(V^*)^2}{1000\lambda}$	1

Таблица 3 – Ручное определение расчётных характеристик программы на Pascal

С помощью программы автоматизации расчёта метрик Холстеда были получены следующие результаты: подсчитаны операторы и операнды в программе на Pascal (см. Таблицу 4), определены измеримые характеристики программы на Pascal (см. Таблицу 5), определены расчётные характеристики программы на Pascal (см. Таблицу 6). Скриншоты результатов работы программы представлены в приложении Б.

№	Оператор	Количество	№	Операнд	Количество
1	()	7	*1*	' '	1
2	+	2	*2*	1	8
3	-	1	*3*	100	1
4	;	32	*4*	7	1
5	=	9	*5*	80	1
6	>	1	*6*	B_sort1	1
7	[]	6	*7*	a	5
8	const	1	*8*	ary	1
9	for	3	*9*	false	1
10	if	1	*10*	hold	3
11	program	1	*11*	i	4
12	random	1	*12*	j	5
13	randomize	1	*13*	max	3
14	repeat	1	*14*	n	7
15	sort	2	*15*	no_change	4
16	swap	2	*16*	p	3
17	type	1	*17*	q	3
18	write	1	*18*	true	1
19	write_arr	1	*19*	x	4
20	writeln	2			

Таблица 4 – Программный расчёт операторов и операндов в программе на Pascal

Характеристика	Формула	Значение
Число уникальных операторов	η_1	20
Число уникальных операндов	η_2	19

Число всех операторов	N_1	76
Число всех операндов	N_2	57
Словарь программы	$\eta = \eta_1 + \eta_2$	39
Длина программы	$N = N_1 + N_2$	133

Таблица 5 – Программное определение измеримых характеристик программы на Pascal

Характеристика	Формула	Значение
Теоретическая оценка длины программы	$\check{N} = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2$	167
Реальный объём	$V = N \log_2 \eta$	702,96
Потенциальный объём	$V^* = (2 + \eta_2^*) \log_2 (2 + \eta_2^*)$	8
Уровень программы	$L = \frac{V^*}{V}$	0,011
Интеллектуальное содержание программы	$I = \frac{2\eta_2}{\eta_1 N_2} (N_1 + N_2) \log_2 (\eta_1 + \eta_2)$	23,43
Работа в программировании	$E = \frac{V}{L}$	61769
Время программирования	$T = \frac{E}{S}$	3432
Уровень языка	$\lambda = LV^*$	0,091
Ожидаемое число ошибок	$B = \frac{(V^*)^2}{1000\lambda}$	1

Таблица 6 – Программное определение расчётных характеристик программы на Pascal

На основе программы на языке Pascal была написана аналогичная программа на языке C. Эта программа представлена в Приложении В. Для этой программы был произведён ручной расчёт операторов и операндов. Результат представлен в Таблице 7.

№	Оператор	Количество	№	Операнд	Количество
1	;	20	*1*	hold	2
2	=	11	*2*	p	2
3	() или {}	31	*3*	q	2
4	if	1	*4*	result	4
5	do … while	1	*5*	no_change	3

6	for	3	*6*	j	5
7	!	1	*7*	n *(в функции sort)*	1
8	%	1	*8*	1	3
9	<	3	*9*	0	5
10	>	1	*10*	i *(в функции write_arr)*	3
11	+	7	*11*	n *(в функции write_arr)*	1
12	-	1	*12*	n *(в функции main)*	2
13	++	3	*13*	max	1
14	* (*умножить*)	2	*14*	i *(в функции main)*	4
15	* (*указатель*)	7	*15*	x	1
16	return	2	*16*	100	1
17	[]	1			
18	void swap	1			
19	double* sort	1			
20	void write_arr	1			
21	int main	1			
22	sizeof	2			
23	malloc(…)	1			
24	memcpy(…)	1			
25	swap(…)	1			
26	printf("\n")	2			
27	printf("%7.1f ", *(a + i))	1			
28	time(NULL)	1			
29	srand(…)	1			
30	rand()	1			
31	sort()	1			

Таблица 7 – Ручной расчёт операторов и операндов в программе на C

Затем был произведён расчёт измеримых характеристик этой программы. Полученный результат представлен в Таблице 8.

Характеристика	**Формула**	**Значение**
Число уникальных операторов	η_1	31
Число уникальных операндов	η_2	16

Число всех операторов	N_1	112
Число всех операндов	N_2	40
Словарь программы	$\eta = \eta_1 + \eta_2$	47
Длина программы	$N = N_1 + N_2$	128

Таблица 8 – Ручное определение измеримых характеристик программы на C

После чего были получены расчётные характеристики программы на C. Результат представлен в Таблице 9.

Для расчётов значение коэффициента Стауда *S* принято 10; значение η_2^* принято 3, поскольку исследуемая функция *sort* принимает 2 аргумента и имеет возвращаемое значение.

Характеристика	**Формула**	**Значение**
Теоретическая оценка длины программы	$\check{\mathrm{N}} = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2$	217,58
Реальный объём	$V = N \log_2 \eta$	844,30
Потенциальный объём	$V^* = (2 + \eta_2^*) \log_2 (2 + \eta_2^*)$	11,61
Уровень программы	$L = \frac{V^*}{V}$	0,014
Интеллектуальное содержание программы	$I = \frac{2\eta_2}{\eta_1 N_2} (N_1 + N_2) \log_2 (\eta_1 + \eta_2)$	21,79
Работа в программировании	$E = \frac{V}{L}$	61401
Время программирования	$T = \frac{E}{S}$	6140
Уровень языка	$\lambda = LV^*$	0,160
Ожидаемое число ошибок	$B = \frac{(V^*)^2}{1000\lambda}$	1

Таблица 9 – Ручное определение расчётных характеристик программы на C

С помощью программы автоматизации расчёта метрик Холстеда были получены следующие результаты: подсчитаны операторы и операнды в программе на C (см. Таблицу 10), определены измеримые характеристики программы на C (см. Таблицу 11), определены расчётные характеристики

программы на С (см. Таблицу 12). Скриншоты результатов работы программы представлены в приложении Г.

№	Оператор	Количество	№	Операнд	Количество
1	!	1	*1*	"%7.1f "	1
2	%	1	*2*	"\n"	2
3	()	10	*3*	0	5
4	*	2	*4*	1	4
5	+	7	*5*	100	1
6	++	3	*6*	NULL	1
7	,	8	*7*	a	4
8	-	1	*8*	hold	2
9	;	32	*9*	i	8
10	<	3	*10*	j	7
11	=	11	*11*	max	2
12	>	1	*12*	n	9
13	sizeof	2	*13*	no_change	4
14	[]	1	*14*	p	3
15	_*	7	*15*	q	3
16	_[]	1	*16*	result	7
17	__*	6	*17*	x	3
18	do while	1			
19	for	3			
20	if	1			
21	main	1			
22	malloc	1			
23	memcpy	1			
24	printf	3			
25	rand	1			
26	return	2			
27	sort	2			
28	srand	1			
29	swap	2			
30	time	1			
31	write_arr	1			

Таблица 10 – Программный расчёт операторов и операндов в программе на С

Характеристика	Формула	Значение
Число уникальных операторов	η_1	31
Число уникальных операндов	η_2	17
Число всех операторов	N_1	118
Число всех операндов	N_2	66
Словарь программы	$\eta = \eta_1 + \eta_2$	48
Длина программы	$N = N_1 + N_2$	184

Таблица 11 – Программное определение измеримых характеристик программы на C

Характеристика	Формула	Значение
Теоретическая оценка длины программы	$\hat{N} = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2$	223
Реальный объём	$V = N \log_2 \eta$	1027,63
Потенциальный объём	$V^* = (2 + \eta_2^*) \log_2 (2 + \eta_2^*)$	11,61
Уровень программы	$L = \frac{V^*}{V}$	0,011
Интеллектуальное содержание программы	$I = \frac{2\eta_2}{\eta_1 N_2} (N_1 + N_2) \log_2 (\eta_1 + \eta_2)$	17,08
Работа в программировании	$E = \frac{V}{L}$	90962
Время программирования	$T = \frac{E}{S}$	5053
Уровень языка	$\lambda = LV^*$	0,131
Ожидаемое число ошибок	$B = \frac{(V^*)^2}{1000\lambda}$	1

Таблица 12 – Программное определение расчётных характеристик программы на C

С помощью команды:

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

Был получен ассемблерный код этой программы в файле *BubbleSort.s*. В полученном коде были удалены все комментарии и отладочные директивы. Итоговый ассемблерный код представлен в приложении Д. Для ассемблерной

программы был произведён ручной расчёт операторов и операндов. Результат представлен в Таблице 13.

№	Оператор	Количество	№	Операнд	Количество
1	push	4	*1*	rbp	9
2	mov	64	*2*	rsp	7
3	movsd	10	*3*	QWORD PTR -24[rbp]	7
4	nop	2	*4*	rdi	8
5	pop	1	*5*	QWORD PTR -32[rbp]	3
6	ret	4	*6*	rsi	5
7	sub	6	*7*	rax	29
8	cdqe	8	*8*	xmm0	11
9	sal	1	*9*	QWORD PTR [rax]	7
10	lea	8	*10*	QWORD PTR -8[rbp]	11
11	jmp .L3	1	*11*	32	1
12	add	10	*12*	DWORD PTR -28[rbp]	6
13	ucomisd	1	*13*	esi	3
14	jbe .L4	1	*14*	eax	31
15	cmp	4	*15*	3	1
16	jl .L6	1	*16*	rdx	10
17	je .L7	1	*17*	0[0+rax*8]	6
18	leave	3	*18*	rcx	4
19	jmp .L11	1	*19*	BYTE PTR -13[rbp]	3
20	jl .L12	1	*20*	1	8
21	xor	2	*21*	DWORD PTR -12[rbp]	7
22	jmp .L14	1	*22*	0	7
23	imul	2	*23*	xmm1	2
24	sar	2	*24*	48	1
25	cvtsi2sd	1	*25*	DWORD PTR -4[rbp]	4
26	jl .L15	1	*26*	QWORD PTR -40[rbp]	2
27	je .L17	1	*27*	.LC0[rip]	1
28	call malloc@PLT	1	*28*	edi	3
29	call memcpy@PLT	1	*29*	10	2
30	call swap	1	*30*	672	1
31	call putchar@PLT	2	*31*	QWORD PTR fs:40	2
32	call printf@PLT	1	*32*	DWORD PTR -660[rbp]	3
33	call time@PLT	1	*33*	80	1
34	call srand@PLT	1	*34*	DWORD PTR -664[rbp]	4

35	call rand@PLT	1	*35*	ecx	5
36	call sort	1	*36*	1374389535	1
37	call__stack_chk_fail@PLT	1	*37*	5	1
			38	31	1
			39	100	1
			40	QWORD PTR -656[rbp+rax*8]	1
			41	-656[rbp]	1

Таблица 13 – Ручной расчёт операторов и операндов в программе на ассемблере

Затем был произведён расчёт измеримых характеристик этой программы. Полученный результат представлен в Таблице 14.

Характеристика	Формула	Значение
Число уникальных операторов	η_1	37
Число уникальных операндов	η_2	41
Число всех операторов	N_1	153
Число всех операндов	N_2	221
Словарь программы	$\eta = \eta_1 + \eta_2$	78
Длина программы	$N = N_1 + N_2$	374

Таблица 14 – Ручное определение измеримых характеристик программы на ассемблере

После чего были получены расчётные характеристики программы на ассемблере. Результат представлен в Таблице 15.

Для расчётов значение коэффициента Стауда S принято 10; значение η_2^* принято 3, поскольку исследуемая функция *sort* принимает 2 аргумента и имеет возвращаемое значение.

Характеристика	Формула	Значение
Теоретическая оценка длины программы	$\hat{N} = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2$	412,41
Реальный объём	$V = N \log_2 \eta$	2350,74
Потенциальный объём	$V^* = (2 + \eta_2^*) \log_2 (2 + \eta_2^*)$	11,61
Уровень программы	$L = \frac{V^*}{V}$	0,005

Интеллектуальное содержание программы	$I = \frac{2\eta_2}{\eta_1 N_2}(N_1 + N_2)\log_2(\eta_1 + \eta_2)$	23,57
Работа в программировании	$E = \frac{V}{L}$	475982
Время программирования	$T = \frac{E}{S}$	47598
Уровень языка	$\lambda = LV^*$	0,057
Ожидаемое число ошибок	$B = \frac{(V^*)^2}{1000\lambda}$	3

Таблица 15 – Ручное определение расчётных характеристик программы на ассемблере

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, а также 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.

ПРИЛОЖЕНИЯ

Приложение А. Исходный код программы на Pascal.

```
program B_sort1;
{ bubble sorting – vers.1 with swap-procedure }

const max = 80;
type ary = array[1..max] of real;
var x: ary;
    i, n: integer;

procedure {bubble} sort(var a: ary; n: integer);
{ adapted from 'Introduction to PASCAL',
      R.Zaks, Sybex, 1980 }

var   no_change   : boolean;
    j: integer;

procedure swap(var p,q: real);
var   hold: real;
begin
  hold:=p;
  p:=q;
  q:=hold;
end;        { swap }

begin       { procedure sort }
  repeat
    no_change:=true;
    for j:=1 to n-1 do
      begin
            if a[j]>a[j+1] then
              begin
               swap(a[j],a[j+1]);
               no_change:=false;
             end
         end     { for }
  until no_change
end;  { procedure sort }

procedure write_arr;
{ print out the answer }
var
  i: integer;

begin
  writeln;
  for i:=1 to n do
      write(x[i]:7:1, ' ');
      writeln;
end;        { write_arr }

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

Приложение Б. Результат работы программы автоматизации расчёта метрик Холстеда в программе на Pascal.

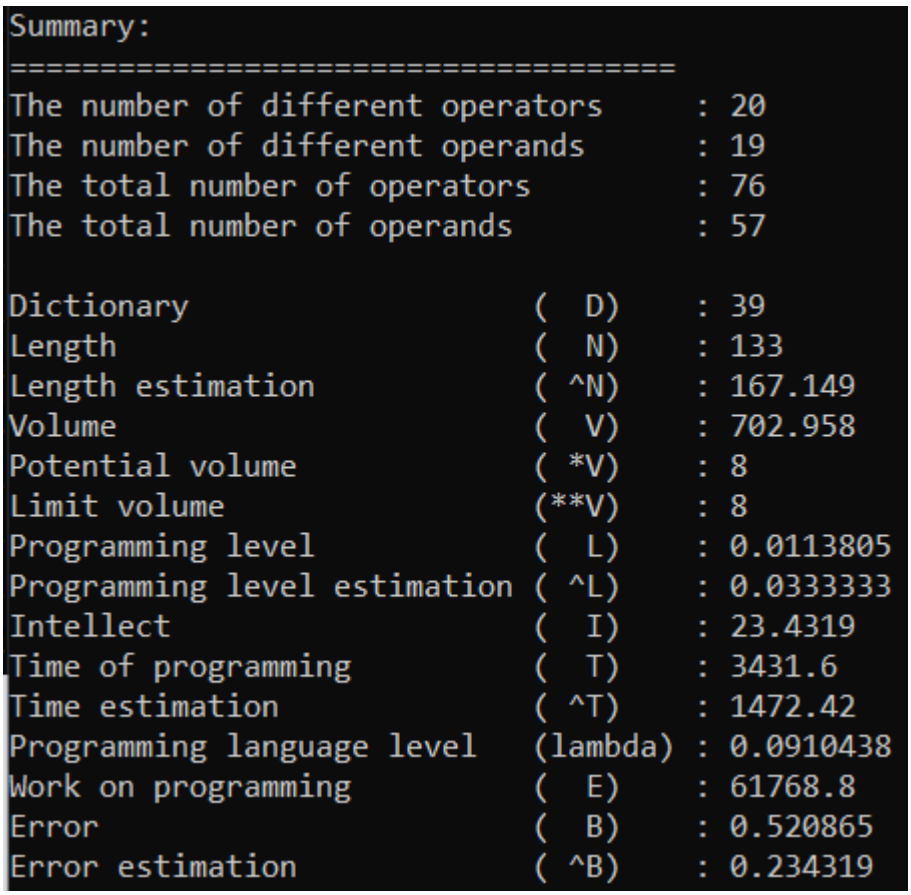
Table:

Operators:

1	7	()
2	2	+
3	1	-
4	32	;
5	9	=
6	1	>
7	6	[]
8	1	const
9	3	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	write
19	1	write_arr
20	2	writeln

Operands:

1	1	' '
2	8	1
3	1	100
4	1	7
5	1	80
6	1	B_sort1
7	5	a
8	1	ary
9	1	false
10	3	hold
11	4	i
12	5	j
13	3	max
14	7	n
15	4	no_change
16	3	p
17	3	q
18	1	true
19	4	x

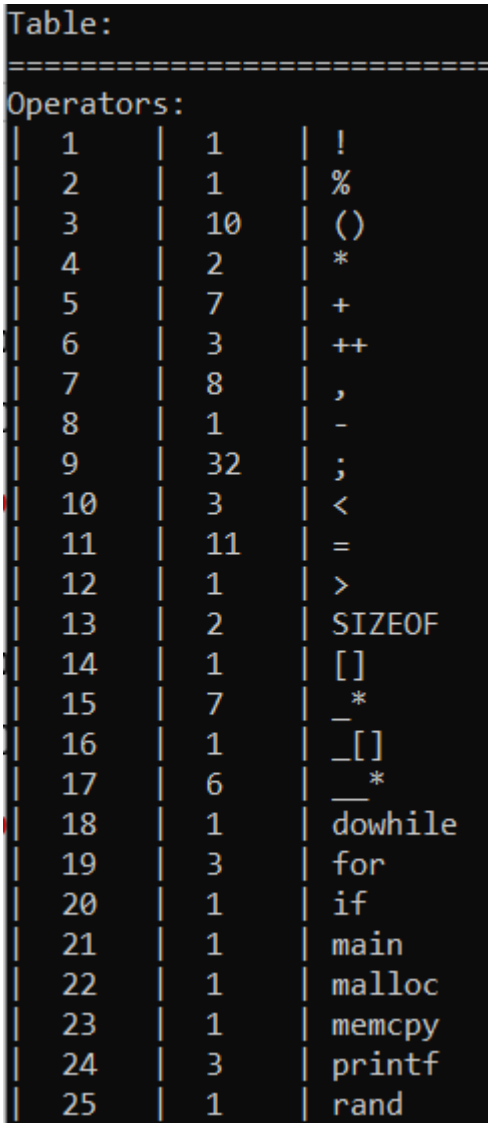


Приложение В. Исходный код программы на С.

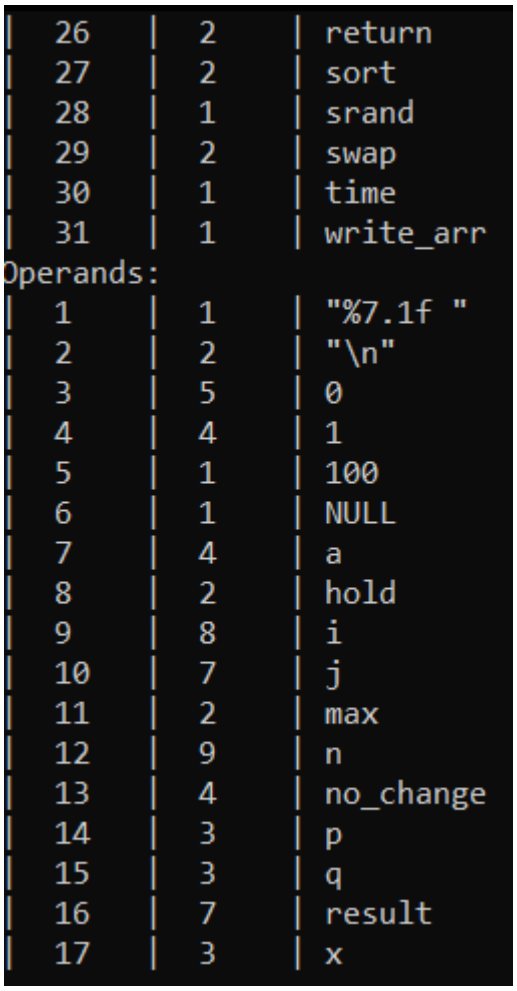
```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>

#define max 80

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

double* sort(double* a, int n)
{
    double* result = malloc(sizeof(double) * n);
    memcpy(result, a, sizeof(double) * n);
    char no_change;
    do
    {
        no_change = 1;
        for (int j = 0; j < n - 1; j++)
        {
            if (*(result + j) > *(result + j + 1))
            {
                swap(result + j, result + j + 1);
                no_change = 0;
            }
```

```
        }
    } while (!no_change);
    return result;
}

void write_arr(double* a, int n)
{
    printf("\n");
    for (int i = 0; i < n; i++)
        printf("%7.1f ", *(a + i));
    printf("\n");
}

int main()
{
    int n = max;
    double x[max];
    srand(time(NULL));
    for (int i = 0; i < n; i++)
        x[i] = rand() % 100;
    sort(x, n);
    return 0;
}
```

Приложение Г. Результат работы программы автоматизации расчёта метрик Холстеда в программе на C.



```
Table:
==========================
Operators:
|  1    |  1    | !
|  2    |  1    | %
|  3    |  10   | ()
|  4    |  2    | *
|  5    |  7    | +
|  6    |  3    | ++
|  7    |  8    | ,
|  8    |  1    | -
|  9    |  32   | ;
|  10   |  3    | <
|  11   |  11   | =
|  12   |  1    | >
|  13   |  2    | SIZEOF
|  14   |  1    | []
|  15   |  7    | _*
|  16   |  1    | _[]
|  17   |  6    | __*
|  18   |  1    | dowhile
|  19   |  3    | for
|  20   |  1    | if
|  21   |  1    | main
|  22   |  1    | malloc
|  23   |  1    | memcpy
|  24   |  3    | printf
|  25   |  1    | rand
```



```
|  26   |  2    | return
|  27   |  2    | sort
|  28   |  1    | srand
|  29   |  2    | swap
|  30   |  1    | time
|  31   |  1    | write_arr
Operands:
|  1    |  1    | "%7.1f "
|  2    |  2    | "\n"
|  3    |  5    | 0
|  4    |  4    | 1
|  5    |  1    | 100
|  6    |  1    | NULL
|  7    |  4    | a
|  8    |  2    | hold
|  9    |  8    | i
|  10   |  7    | j
|  11   |  2    | max
|  12   |  9    | n
|  13   |  4    | no_change
|  14   |  3    | p
|  15   |  3    | q
|  16   |  7    | result
|  17   |  3    | x
```

```
Summary:
====================================
The number of different operators     : 31
The number of different operands      : 17
The total number of operators         : 118
The total number of operands          : 66

Dictionary                    (  D)   : 48
Length                        (  N)   : 184
Length estimation             ( ^N)   : 223.067
Volume                        (  V)   : 1027.63
Potential volume              ( *V)   : 11.6096
Limit volume                  (**V)   : 15.6844
Programming level             (  L)   : 0.0112975
Programming level estimation  ( ^L)   : 0.0166178
Intellect                     (  I)   : 17.077
Time of programming           (  T)   : 5053.41
Time estimation               ( ^T)   : 4164.95
Programming language level    (lambda) : 0.131159
Work on programming           (  E)   : 90961.5
Error                         (  B)   : 0.674192
Error estimation              ( ^B)   : 0.342544
```

Приложение Д. Ассемблерный код программы.

```
        .intel_syntax noprefix
swap:
        push  rbp
        mov   rbp, rsp
        mov   QWORD PTR -24[rbp], rdi
        mov   QWORD PTR -32[rbp], rsi
        mov   rax, QWORD PTR -24[rbp]
        movsd xmm0, QWORD PTR [rax]
        movsd QWORD PTR -8[rbp], xmm0
        mov   rax, QWORD PTR -32[rbp]
        movsd xmm0, QWORD PTR [rax]
        mov   rax, QWORD PTR -24[rbp]
        movsd QWORD PTR [rax], xmm0
        mov   rax, QWORD PTR -32[rbp]
        movsd xmm0, QWORD PTR -8[rbp]
        movsd QWORD PTR [rax], xmm0
        nop
        pop   rbp
        ret
sort:
        push  rbp
        mov   rbp, rsp
        sub   rsp, 32
        mov   QWORD PTR -24[rbp], rdi
        mov   DWORD PTR -28[rbp], esi
        mov   eax, DWORD PTR -28[rbp]
        cdqe
        sal   rax, 3
        mov   rdi, rax
```

```
      call  malloc@PLT
      mov   QWORD PTR -8[rbp], rax
      mov   eax, DWORD PTR -28[rbp]
      cdqe
      lea   rdx, 0[0+rax*8]
      mov   rcx, QWORD PTR -24[rbp]
      mov   rax, QWORD PTR -8[rbp]
      mov   rsi, rcx
      mov   rdi, rax
      call  memcpy@PLT
.L7:
      mov   BYTE PTR -13[rbp], 1
      mov   DWORD PTR -12[rbp], 0
      jmp   .L3
.L6:
      mov   eax, DWORD PTR -12[rbp]
      cdqe
      lea   rdx, 0[0+rax*8]
      mov   rax, QWORD PTR -8[rbp]
      add   rax, rdx
      movsd xmm0, QWORD PTR [rax]
      mov   eax, DWORD PTR -12[rbp]
      cdqe
      add   rax, 1
      lea   rdx, 0[0+rax*8]
      mov   rax, QWORD PTR -8[rbp]
      add   rax, rdx
      movsd xmm1, QWORD PTR [rax]
      ucomisd     xmm0, xmm1
      jbe   .L4
      mov   eax, DWORD PTR -12[rbp]
      cdqe
      add   rax, 1
      lea   rdx, 0[0+rax*8]
      mov   rax, QWORD PTR -8[rbp]
      add   rdx, rax
      mov   eax, DWORD PTR -12[rbp]
      cdqe
      lea   rcx, 0[0+rax*8]
      mov   rax, QWORD PTR -8[rbp]
      add   rax, rcx
      mov   rsi, rdx
      mov   rdi, rax
      call  swap
      mov   BYTE PTR -13[rbp], 0
.L4:
      add   DWORD PTR -12[rbp], 1
.L3:
      mov   eax, DWORD PTR -28[rbp]
      sub   eax, 1
      cmp   DWORD PTR -12[rbp], eax
      jl    .L6
      cmp   BYTE PTR -13[rbp], 0
      je    .L7
      mov   rax, QWORD PTR -8[rbp]
      leave
      ret
.LC0:
      .string     "%7.1f "
write_arr:
      push  rbp
      mov   rbp, rsp
      sub   rsp, 48
```

```
        mov   QWORD PTR -24[rbp], rdi
        mov   DWORD PTR -28[rbp], esi
        mov   edi, 10
        call  putchar@PLT
        mov   DWORD PTR -4[rbp], 0
        jmp   .L11
.L12:
        mov   eax, DWORD PTR -4[rbp]
        cdqe
        lea   rdx, 0[0+rax*8]
        mov   rax, QWORD PTR -24[rbp]
        add   rax, rdx
        mov   rax, QWORD PTR [rax]
        mov   QWORD PTR -40[rbp], rax
        movsd xmm0, QWORD PTR -40[rbp]
        lea   rdi, .LC0[rip]
        mov   eax, 1
        call  printf@PLT
        add   DWORD PTR -4[rbp], 1
.L11:
        mov   eax, DWORD PTR -4[rbp]
        cmp   eax, DWORD PTR -28[rbp]
        jl    .L12
        mov   edi, 10
        call  putchar@PLT
        nop
        leave
        ret

        .globl      main
main:
        push  rbp
        mov   rbp, rsp
        sub   rsp, 672
        mov   rax, QWORD PTR fs:40
        mov   QWORD PTR -8[rbp], rax
        xor   eax, eax
        mov   DWORD PTR -660[rbp], 80
        mov   edi, 0
        call  time@PLT
        mov   edi, eax
        call  srand@PLT
        mov   DWORD PTR -664[rbp], 0
        jmp   .L14
.L15:
        call  rand@PLT
        mov   ecx, eax
        mov   edx, 1374389535
        mov   eax, ecx
        imul  edx
        sar   edx, 5
        mov   eax, ecx
        sar   eax, 31
        sub   edx, eax
        mov   eax, edx
        imul  eax, eax, 100
        sub   ecx, eax
        mov   eax, ecx
        cvtsi2sd    xmm0, eax
        mov   eax, DWORD PTR -664[rbp]
        cdqe
        movsd QWORD PTR -656[rbp+rax*8], xmm0
        add   DWORD PTR -664[rbp], 1
```

```
.L14:
      mov   eax, DWORD PTR -664[rbp]
      cmp   eax, DWORD PTR -660[rbp]
      jl    .L15
      mov   edx, DWORD PTR -660[rbp]
      lea   rax, -656[rbp]
      mov   esi, edx
      mov   rdi, rax
      call  sort
      mov   eax, 0
      mov   rsi, QWORD PTR -8[rbp]
      xor   rsi, QWORD PTR fs:40
      je    .L17
      call  __stack_chk_fail@PLT
.L17:
      leave
      ret
```