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 по метрикам Холстеда»

Студентка гр. 6304 _____ Иванкова В.М.

Преподаватель _____ Кирьянчиков В.А.

Санкт-Петербург
2020

Задание.

Для заданного варианта программы обработки данных, представленной на языке Паскаль, разработать вычислительный алгоритм и варианты программ его реализации на языках программирования Си и Ассемблер. Добиться, чтобы программы на Паскале и Си были работоспособны и давали корректные результаты (это потребуется в дальнейшем при проведении с ними измерительных экспериментов). Для получения ассемблерного представления программы можно либо самостоятельно написать код на ассемблере, реализующий заданный алгоритм, либо установить опцию "Codegeneration/Generateassemblersource" при компиляции текста программы, представленной на языке Си. Во втором случае в ассемблерном представлении программы нужно удалить директивы описаний и отладочные директивы, оставив только исполняемые операторы.

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, потенциальный и граничный объемы программы;
- уровень программы;
- интеллектуальное содержание программы;

- работу программиста;
- время программирования;
- уровень используемого языка программирования;
- ожидаемое число ошибок в программе.

Для каждой характеристики следует рассчитать, как саму характеристику, так и ее оценку.

Расчет характеристик программ и их оценок выполнить двумя способами:

1)вручную (с калькулятором) или с помощью одного из доступных средств математических вычислений EXCEL, MATHCAD или MATLAB.

2) с помощью программы автоматизации расчета метрик Холстеда (для С- и Паскаль-версий программ), краткая инструкция по работе, с которой приведена в файле user_guide.

Для варианта расчета с использованием программы автоматизации желательно провести анализ влияния учета тех или иных групп операторов исследуемой программы на вычисляемые характеристики за счет задания разных ключей запуска.

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

Результаты расчетов представить в виде сводных таблиц с текстовыми комментариями.

Расчет метрик вручную

Программа на языке Паскаль, С и Assembler представлены в приложениях А, Б и В, соответственно.

В таблицах 1-3 представлены результаты подсчета числа типов операторов и операндов в программах на языке Паскаль, С и Assembler.

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

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	;	14	1	80	1
2	begin... end	7	2	max	3
3	:=	9	3	x	4
4	for…to…do	4	4	i	11
5	if…then	2	5	j	5
6	repeat…until	1	6	n	8
7	+	4	7	hold	3
8	while…end	1	8	a	5
9	>	2	9	jump	6
10	[]	15	10	p	3
11	div	1	11	q	3
12	swap	1	12	1	5
13	sort	1	13	2	1
14	write_arr	1	14	7	1
15	random	1	15	0	1
16	randomize	1	16	false	1
17	()	3	17	true	1
			18	done	4
			19	100	1

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

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	shellsort	1	1	100.0	1
2	swap	1	2	i	15
3	write_arr	1	3	j	5
4	break	1	4	k	14
5	for	5	5	temp	3
6	if…else	1	6	arr	11
7	[]	7	7	num	10
8	=	11	8	my_max	2

9	/	4	9	RAND_MAX	1
10	+	2	10	80	1
11	-	2	11	0	6
12	<	3	12	2	2
13	>	1	13	x	3
14	>=	2	14	y	3
15	;	25			
16	return	1			
17	*	4			
18	rand	1			
19	++	3			
20	&	2			
21	()	14			

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

№	Оператор	Число вхождений	№	Операнд	Число вхождений
1	pushq	5	1	-24(%rbp)	12
2	movq	36	2	-32(%rbp)	7
3	subq	6	3	-4(%rbp)	13
4	xorl	1	4	32	3
5	movl	39	5	-28(%rbp)	5
6	movss	13	6	31	2
7	movslq	4	7	-12(%rbp)	9
8	cltq	6	8	-8(%rbp)	5
9	nop	4	9	0(,%rax,4)	5
10	popq	1	10	0	8
11	ret	4	11	1	6
12	shrl	2	12	56	1
13	addl	7	13	80	1
14	sarl	2	14	-48(%rbp)	5
15	jmp	6	15	-44(%rbp)	2
16	leaq	7	16	-40(%rbp)	1
17	subl	2	17	2	3
18	addq	7	18	3(%rax)	1
19	ucomiss	1	19	16	3
20	jnb	1	20	3	1
21	cmpl	5	21	-52(%rbp)	4
22	jns	1	22	(%rax,%rdx,4)	1
23	jl	3	23	rbp	9
24	jg	1	24	rsp	11
25	leave	3	25	rdi	7
26	cvtss2sd	1	26	rsi	3
27	call	6	27	rax	42

28	salq	2	28	xmm0	16
29	divq	1	29	esi	4
30	imulq	1	30	eax	40
31	shrq	1	31	edx	17
32	cvtsi2ss	1	32	rdx	19
33	divss	2	33	xmm1	5
34	xorq	1	34	rcx	5
35	je	1	35	rip	3
36			36	rbx	4
37			37	r8	1
38			38	r9d	1
39			39	edi	1
40			40	ecx	1
41			41	fs:40	2

В таблице 4 представлены сводные результаты расчетных характеристик вручную.

Таблица 4 – Результаты расчетных характеристик вручную

	Паскаль	**Си**	**Ассемблер**
Число уникальных операторов (n1):	17	21	35
Число уникальных операндов (n2):	19	14	41
Общее число операторов (N1):	68	92	184
Общее число операндов (N2):	67	77	286
Алфавит (n):	36	35	76
Экспериментальная длина программы (Nэ):	135	169	470
Теоретическая длина программы (Nт):	150,1975	145,5416	399,1845
Объём программы (V):	697,9399	866,84883	2936,5259
Потенциальный объём (V*):	11,6	11,6	11,6
Уровень программы (L):	0,01662	0,01338	0,00395
Сложность программы (S):	60,16847	74,738415	253,16456
Ожидание уровня программы (L^):	0,03336	0,017316	0,00819
Интеллект программы (I):	23,28327	15,010354	24,05014
Работа по программированию (E):	41993,9771	64786,90807	743424,278
Время кодирования (T):	2091,98294	5006,051992	35847,10314

Ожидание времени кодирования (T^):	4199,39771	6478,69081	74342,4278
Уровень языка программирования (Lam):	0,192792	0,2008656	0,095004
Уровень ошибок (B):	1	1	2

Расчет метрик с помощью программы автоматизации

Для программы на Паскале:

```
Statistics for module lab_1_pasc_out.lxm
=====================================
The number of different operators     : 25
The number of different operands      : 21
The total number of operators         : 107
The total number of operands          : 67

Dictionary                    (  D)   : 46
Length                        (  N)   : 174
Length estimation             ( ^N)   : 208.335
Volume                        (  V)   : 961.1
Potential volume              ( *V)   : 19.6515
Limit volume                  (**V)   : 38.2071
Programming level             (  L)   : 0.0204469
Programming level estimation  ( ^L)   : 0.0250746
Intellect                     (  I)   : 24.0992
Time of programming           (  T)   : 2611.37
Time estimation               ( ^T)   : 2549.62
Programming language level    (lambda) : 0.401811
Work on programming           (  E)   : 47004.7
Error                         (  B)   : 0.43415
Error estimation              ( ^B)   : 0.320367


Table:
=====================================
Operators:
|  1|  11    | ()
|  2|  1| +
|  3|  1| /
|  4|  42    | ;
|  5|  11    | =
|  6|  3| >
|  7|  6| []
```

```
|  8|  1| and
|  9|  1| boolean
|  10   |  1| const
|  11   |  3| for
|  12   |  1| if
|  13   |  4| integer
|  14   |  3| procedure
|  15   |  1| program
|  16   |  1| random
|  17   |  1| randomize
|  18   |  4| real
|  19   |  1| repeat
|  20   |  2| sort
|  21   |  2| swap
|  22   |  1| while
|  23   |  1| write
|  24   |  2| write_arr
|  25   |  2| writeln
Operands:
|  1|  1| ' '
|  2|  1| 0
|  3|  5| 1
|  4|  1| 100
|  5|  1| 2
|  6|  1| 7
|  7|  1| 80
|  8|  1| Shell_sort
|  9|  5| a
|  10   |  4| done
|  11   |  1| false
|  12   |  3| hold
|  13   |  9| i
|  14   |  4| j
|  15   |  6| jump
|  16   |  3| max
|  17   |  9| n
|  18   |  3| p
|  19   |  3| q
|  20   |  1| true
|  21   |  4| x


Summary:
====================================
The number of different operators     : 25
```

```
The number of different operands      : 21
The total number of operators         : 107
The total number of operands          : 67

Dictionary                    (  D)   : 46
Length                        (  N)   : 174
Length estimation             ( ^N)   : 208.335
Volume                        (  V)   : 961.1
Potential volume              ( *V)   : 19.6515
Limit volume                  (**V)   : 38.2071
Programming level             (  L)   : 0.0204469
Programming level estimation  ( ^L)   : 0.0250746
Intellect                     (  I)   : 24.0992
Time of programming           (  T)   : 2611.37
Time estimation               ( ^T)   : 2549.62
Programming language level    (lambda) : 0.401811
Work on programming           (  E)   : 47004.7
Error                         (  B)   : 0.43415
Error estimation              ( ^B)   : 0.320367
```

Для программы на Си:

```
Statistics for module lab_1_c_out.lxm
=====================================
The number of different operators     : 29
The number of different operands      : 15
The total number of operators         : 142
The total number of operands          : 77

Dictionary                    (  D)   : 44
Length                        (  N)   : 219
Length estimation             ( ^N)   : 199.485
Volume                        (  V)   : 1195.62
Potential volume              ( *V)   : 19.6515
Limit volume                  (**V)   : 38.2071
Programming level             (  L)   : 0.0164363
Programming level estimation  ( ^L)   : 0.0134348
Intellect                     (  I)   : 16.0629
Time of programming           (  T)   : 4041.25
Time estimation               ( ^T)   : 4503.52
Programming language level    (lambda) : 0.322998
Work on programming           (  E)   : 72742.4
```

```
Error                          (  B)    : 0.580859
Error estimation               ( ^B)    : 0.398539


Table:
====================================
Operators:
|  1|  13   | ()
|  2|  2| +
|  3|  3| ++
|  4|  9| ,
|  5|  2| -
|  6|  4| /
|  7|  37   | ;
|  8|  3| <
|  9|  13   | =
|  10   |  1| >
|  11   |  2| >=
|  12   |  6| []
|  13   |  2| _&
|  14   |  4| _*
|  15   |  3| _[]
|  16   |  2| __*
|  17   |  1| break
|  18   |  9| float
|  19   |  5| for
|  20   |  1| if
|  21   |  7| int
|  22   |  1| main
|  23   |  1| printf
|  24   |  1| rand
|  25   |  1| return
|  26   |  2| shellsort
|  27   |  2| swap
|  28   |  3| void
|  29   |  2| write_arr
Operands:
|  1|  1| "%f "
|  2|  5| 0
|  3|  1| 100.0
|  4|  2| 2
|  5|  1| 80
|  6|  1| RAND_MAX
|  7|  11   | arr
|  8|  15   | i
```

```
|   9|   5| j
|   10    |   14    | k
|   11    |   2| my_max
|   12    |   10    | num
|   13    |   3| temp
|   14    |   3| x
|   15    |   3| y


Summary:
====================================
The number of different operators     : 29
The number of different operands      : 15
The total number of operators         : 142
The total number of operands          : 77

Dictionary                    (  D)    : 44
Length                        (  N)    : 219
Length estimation             ( ^N)    : 199.485
Volume                        (  V)    : 1195.62
Potential volume              ( *V)    : 19.6515
Limit volume                  (**V)    : 38.2071
Programming level             (  L)    : 0.0164363
Programming level estimation  ( ^L)    : 0.0134348
Intellect                     (  I)    : 16.0629
Time of programming           (  T)    : 4041.25
Time estimation               ( ^T)    : 4503.52
Programming language level    (lambda) : 0.322998
Work on programming           (  E)    : 72742.4
Error                         (  B)    : 0.580859
Error estimation              ( ^B)    : 0.398539
```

Таблица 5 – Сводная таблица расчетов на двух языках

	Паскаль	**Программный расчет паскаль**	**Си**	**Программный расчет Си**
Число уникальных операторов (n1):	17	25	21	29
Число уникальных операндов (n2):	19	21	14	15
Общее число операторов (N1):	68	107	92	142

Общее число операндов (N2):	67	67	77	77
Алфавит (n):	36	46	35	44
Экспериментальная длина программы (Nэ):	135	174	169	219
Теоретическая длина программы (Nт):	150,1975	208,335	145,5416	199,485
Объём программы (V):	697,9399	961,1	866,84883	1195,62
Потенциальный объём (V*):	11,6	19,6515	11,6	19,6515
Уровень программы (L):	0,01662	0,0204469	0,01338	0,0164363
Ожидание уровня программы (L^):	0,03336	0,0250746	0,017316	0,0134348
Интеллект программы (I):	23,28327	24,0992	15,010354	16,0629
Работа по программированию (E):	41993,9771	47004,7	64786,90807	72742,4
Время кодирования (T):	2091,98294	2611,37	5006,051992	4041,25
Ожидание времени кодирования (T^):	4199,39771	2549,62	6478,69081	4503,52
Уровень языка программирования (Lam):	0,192792	0,401811	0,2008656	0,322998
Уровень ошибок (B):	1	1	1	1

Вывод

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

Все характеристики были посчитаны вручную и автоматически. Различия между методами присутствует из-за того, что программа считает не только функциональную часть, но и объявления типов, переменных и функций.

ПРИЛОЖЕНИЕ А. КОД ПРОГРАММЫ НА ЯЗЫКЕ ПАСКАЛЬ

```
program Shell_sort;
const   max = 80;
var x: array[1..max]of real;i,n: integer;

procedure {shell} sort(var a: array of real; 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;    { swap }

begin
  jump:=n;
  while jump>1 do
    begin
      jump:=jump div 2;
      repeat
  done:=true;
  for j:=0 to n do
    begin
      i:=j+jump;
      if (n>i) and (a[j]>a[i]) then
        begin
    swap(a[j],a[i]);
    done:=false;
      end;    { if }
  end;    { for }
      until done;
  end     { while }
end;    { 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
    begin
   x[i]:= random(100);
   end;
 sort( x,n );
 write_arr;
end.
```

ПРИЛОЖЕНИЕ Б. ПРОГРАММА НА ЯЗЫКЕ СИ

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

void swap (float *x, float *y)
{
    float temp;
    temp = *x;
    *x = *y;
    *y = temp;
}

void shellsort(float arr[], int num)
{
    int i, j, k;
    for (i = num / 2; i > 0; i = i / 2)
    {
        for (j = i; j < num; j++)
        {
            for(k = j - i; k >= 0; k = k - i)
            {
                if (arr[k+i] >= arr[k])
                    break;
                else
                {
                    swap(&arr[k], &arr[k+i]);
                }
            }
        }
    }
}

void write_arr(float arr[], int num)
{
    int i;
    for (i = 0; i < num; i++)
    {
        printf("%f ", arr[i]);
    }
}
int main()
{
    int num = 80;
    float my_max = 100.0;
    float arr[num];
    int k;

    for (k = 0 ; k < num; k++)
    {
        arr[k] = (float)rand()/(float)(RAND_MAX/my_max);
    }
    shellsort(arr, num);
```

```
    write_arr(arr, num);
    return 0;
}
```

ПРИЛОЖЕНИЕ В. ПРОГРАММА НА ЯЗЫКЕ АССЕМБЛЕР

```
            .file   "lab_1_pasc.c"
    .text
    .globl  swap
    .type   swap, @function
swap:
.LFB5:
    .cfi_startproc
    pushq   %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset 6, -16
    movq %rsp, %rbp
    .cfi_def_cfa_register 6
    movq %rdi, -24(%rbp)
    movq %rsi, -32(%rbp)
    movq -24(%rbp), %rax
    movss   (%rax), %xmm0
    movss   %xmm0, -4(%rbp)
    movq -32(%rbp), %rax
    movss   (%rax), %xmm0
    movq -24(%rbp), %rax
    movss   %xmm0, (%rax)
    movq -32(%rbp), %rax
    movss   -4(%rbp), %xmm0
    movss   %xmm0, (%rax)
    nop
    popq %rbp
    .cfi_def_cfa 7, 8
    ret
    .cfi_endproc
.LFE5:
    .size   swap, .-swap
    .globl  shellsort
    .type   shellsort, @function
shellsort:
.LFB6:
    .cfi_startproc
    pushq   %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset 6, -16
    movq %rsp, %rbp
    .cfi_def_cfa_register 6
    subq $32, %rsp
    movq %rdi, -24(%rbp)
    movl %esi, -28(%rbp)
    movl -28(%rbp), %eax
    movl %eax, %edx
    shrl $31, %edx
    addl %edx, %eax
    sarl %eax
    movl %eax, -12(%rbp)
    jmp  .L3
```

```
.L11:
    movl -12(%rbp), %eax
    movl %eax, -8(%rbp)
    jmp  .L4
.L10:
    movl -8(%rbp), %eax
    subl -12(%rbp), %eax
    movl %eax, -4(%rbp)
    jmp  .L5
.L9:
    movl -4(%rbp), %edx
    movl -12(%rbp), %eax
    addl %edx, %eax
    cltq
    leaq 0(,%rax,4), %rdx
    movq -24(%rbp), %rax
    addq %rdx, %rax
    movss     (%rax), %xmm0
    movl -4(%rbp), %eax
    cltq
    leaq 0(,%rax,4), %rdx
    movq -24(%rbp), %rax
    addq %rdx, %rax
    movss     (%rax), %xmm1
    ucomiss   %xmm1, %xmm0
    jnb  .L12
    movl -4(%rbp), %edx
    movl -12(%rbp), %eax
    addl %edx, %eax
    cltq
    leaq 0(,%rax,4), %rdx
    movq -24(%rbp), %rax
    addq %rax, %rdx
    movl -4(%rbp), %eax
    cltq
    leaq 0(,%rax,4), %rcx
    movq -24(%rbp), %rax
    addq %rcx, %rax
    movq %rdx, %rsi
    movq %rax, %rdi
    call swap
    movl -12(%rbp), %eax
    subl %eax, -4(%rbp)
.L5:
    cmpl $0, -4(%rbp)
    jns  .L9
    jmp  .L8
.L12:
    nop
.L8:
    addl $1, -8(%rbp)
.L4:
    movl -8(%rbp), %eax
```

```
    cmpl -28(%rbp), %eax
    jl   .L10
    movl -12(%rbp), %eax
    movl %eax, %edx
    shrl $31, %edx
    addl %edx, %eax
    sarl %eax
    movl %eax, -12(%rbp)
.L3:
    cmpl $0, -12(%rbp)
    jg   .L11
    nop
    leave
    .cfi_def_cfa 7, 8
    ret
    .cfi_endproc
.LFE6:
    .size     shellsort, .-shellsort
    .section  .rodata
.LC0:
    .string   "%f "
    .text
    .globl    write_arr
    .type     write_arr, @function
write_arr:
.LFB7:
    .cfi_startproc
    pushq     %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset 6, -16
    movq %rsp, %rbp
    .cfi_def_cfa_register 6
    subq $32, %rsp
    movq %rdi, -24(%rbp)
    movl %esi, -28(%rbp)
    movl $0, -4(%rbp)
    jmp  .L14
.L15:
    movl -4(%rbp), %eax
    cltq
    leaq 0(,%rax,4), %rdx
    movq -24(%rbp), %rax
    addq %rdx, %rax
    movss     (%rax), %xmm0
    cvtss2sd  %xmm0, %xmm0
    leaq .LC0(%rip), %rdi
    movl $1, %eax
    call printf@PLT
    addl $1, -4(%rbp)
.L14:
    movl -4(%rbp), %eax
    cmpl -28(%rbp), %eax
    jl   .L15
```

```
    nop
    leave
    .cfi_def_cfa 7, 8
    ret
    .cfi_endproc
.LFE7:
    .size     write_arr, .-write_arr
    .globl    main
    .type     main, @function
main:
.LFB8:
    .cfi_startproc
    pushq     %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset 6, -16
    movq %rsp, %rbp
    .cfi_def_cfa_register 6
    pushq     %rbx
    subq $56, %rsp
    .cfi_offset 3, -24
    movq %fs:40, %rax
    movq %rax, -24(%rbp)
    xorl %eax, %eax
    movq %rsp, %rax
    movq %rax, %rbx
    movl $80, -48(%rbp)
    movss     .LC1(%rip), %xmm0
    movss     %xmm0, -44(%rbp)
    movl -48(%rbp), %eax
    movslq    %eax, %rdx
    subq $1, %rdx
    movq %rdx, -40(%rbp)
    movslq    %eax, %rdx
    movq %rdx, %r8
    movl $0, %r9d
    movslq    %eax, %rdx
    movq %rdx, %rsi
    movl $0, %edi
    cltq
    salq $2, %rax
    leaq 3(%rax), %rdx
    movl $16, %eax
    subq $1, %rax
    addq %rdx, %rax
    movl $16, %ecx
    movl $0, %edx
    divq %rcx
    imulq     $16, %rax, %rax
    subq %rax, %rsp
    movq %rsp, %rax
    addq $3, %rax
    shrq $2, %rax
    salq $2, %rax
```

```
    movq %rax, -32(%rbp)
    movl $0, -52(%rbp)
    jmp  .L17
.L18:
    call rand@PLT
    cvtsi2ss  %eax, %xmm0
    movss     .LC2(%rip), %xmm1
    divss     -44(%rbp), %xmm1
    divss     %xmm1, %xmm0
    movq -32(%rbp), %rax
    movl -52(%rbp), %edx
    movslq    %edx, %rdx
    movss     %xmm0, (%rax,%rdx,4)
    addl $1, -52(%rbp)
.L17:
    movl -52(%rbp), %eax
    cmpl -48(%rbp), %eax
    jl   .L18
    movq -32(%rbp), %rax
    movl -48(%rbp), %edx
    movl %edx, %esi
    movq %rax, %rdi
    call shellsort
    movq -32(%rbp), %rax
    movl -48(%rbp), %edx
    movl %edx, %esi
    movq %rax, %rdi
    call write_arr
    movl $0, %eax
    movq %rbx, %rsp
    movq -24(%rbp), %rcx
    xorq %fs:40, %rcx
    je   .L20
    call __stack_chk_fail@PLT
.L20:
    movq -8(%rbp), %rbx
    leave
    .cfi_def_cfa 7, 8
    ret
    .cfi_endproc
.LFE8:
    .size     main, .-main
    .section  .rodata
    .align 4
.LC1:
    .long     1120403456
    .align 4
.LC2:
    .long     1325400064
    .ident    "GCC: (Ubuntu 7.4.0-1ubuntu1~18.04.1) 7.4.0"
    .section  .note.GNU-stack,"",@progbits
```