**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»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Путьков Д.А. |
| 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 (c 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 A, 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 | 1000 | 1 |
| 2 | begin... end | 7 | 2 | max | 5 |
| 3 | := | 9 | 3 | x | 1 |
| 4 | for…to…do | 4 | 4 | i | 10 |
| 5 | if…then | 2 | 5 | j | 11 |
| 6 | repeat…until | 1 | 6 | n | 6 |
| 7 | + | 4 | 7 | hold | 6 |
| 8 | - | 2 | 8 | a | 16 |
| 9 | > | 2 | 9 | no\_change | 4 |
| 10 | [] | 15 | 10 | p | 3 |
| 11 | \* | 1 | 11 | q | 3 |
| 12 | swap | 1 | 12 | forFirstSort | 3 |
| 13 | sort1 | 1 | 13 | forSecondSort | 3 |
| 14 | sort2 | 1 | 14 | randomNum | 4 |
| 15 | random | 1 | 15 | false | 1 |
| 16 | randomize | 1 | 16 | true | 1 |
| 17 | () | 3 | 17 | 1 | 9 |

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Оператор** | **Число вхождений** | **№** | **Операнд** | **Число вхождений** |
| 1 | <> | 2 | 1 | 1000 | 1 |
| 2 | () | 16 | 2 | 1 | 7 |
| 3 | = | 22 | 3 | MAX | 6 |
| 4 | ; | 17 | 4 | hold | 5 |
| 5 | for | 4 | 5 | a | 16 |
| 6 | if | 2 | 6 | n | 5 |
| 7 | > | 2 | 7 | i | 18 |
| 8 | [] | 15 | 8 | j | 6 |
| 9 | \* | 8 | 9 | 0 | 7 |
| 10 | % | 1 | 10 | p | 3 |
| 11 | do while | 1 | 11 | q | 3 |
| 12 | + | 3 | 12 | noChange | 4 |
| 13 | - | 1 | 13 | forFirstSort | 3 |
| 14 | ++ | 4 | 14 | forSecondSort | 3 |
| 15 | != | 1 | 15 | randomNumber | 2 |
| 16 | calloc | 2 | 16 | doubleRandomNumber | 4 |
| 17 | rand | 1 | 17 | NULL | 1 |
| 18 | returandomNumber | 1 |  |  |  |
| 19 | swap | 1 |  |  |  |
| 20 | sor1 | 1 |  |  |  |
| 21 | sort2 | 1 |  |  |  |
| 22 | srand | 1 |  |  |  |
| 23 | time | 1 |  |  |  |
| 24 | sizeof | 2 |  |  |  |

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Оператор** | **Число вхождений** | **№** | **Операнд** | **Число вхождений** |
| 1 | pushl | 5 | 1 | 16 | 2 |
| 2 | movl | 73 | 2 | 0 | 9 |
| 3 | subl | 7 | 3 | 1 | 15 |
| 4 | addl | 21 | 4 | 2 | 1 |
| 5 | leal | 14 | 5 | 32 | 1 |
| 6 | fldl | 12 | 6 | -16 | 1 |
| 7 | fxch | 2 | 7 | 48 | 1 |
| 8 | fucomip | 2 | 8 | 8 | 2 |
| 9 | fstp | 2 | 9 | 10 | 4 |
| 10 | fstpl | 10 | 10 | 274877907 | 1 |
| 11 | cmpl | 5 | 11 | 6 | 1 |
| 12 | set | 3 | 12 | 31 | 1 |
| 13 | setg | 1 | 13 | 1000 | 1 |
| 14 | setne | 1 | 14 | 9 | 1 |
| 15 | setle | 1 | 15 | st | 8 |
| 16 | testb | 5 | 16 | al | 15 |
| 17 | leave | 4 | 17 | ebp | 8 |
| 18 | ret | 4 | 18 | esp | 12 |
| 19 | fldz | 1 | 19 | eax | 83 |
| 20 | imull | 2 | 20 | edx | 39 |
| 21 | sar | 2 | 21 | ecx | 8 |
| 22 | fild | 1 | 22 | ebx | 5 |
| 23 | jmp L2 | 1 | 23 | -4(%ebp) | 11 |
| 24 | jmp L3 | 1 | 24 | -8(%ebp) | 15 |
| 25 | jbe L4 | 1 | 25 | 0(,%eax,8) | 12 |
| 26 | jne L6 | 1 | 26 | 8(%ebp) | 28 |
| 27 | jne L7 | 1 | 27 | -16(%ebp) | 2 |
| 28 | jmp L11 | 1 | 28 | 0(,%edx,8) | 2 |
| 29 | jbe L12 | 1 | 29 | 12(%ebp) | 5 |
| 30 | jne L14 | 1 | 30 | 8(%esp) | 1 |
| 31 | jne L15 | 1 | 31 | 4(%esp) | 14 |
| 32 | jmp L18 | 1 | 32 | 40(%esp) | 3 |
| 33 | jne L19 | 1 | 33 | 36(%esp) | 3 |
| 34 | call \_\_\_main | 1 | 34 | 24(%esp) | 4 |
| 35 | call \_calloc | 2 | 35 | 44(%esp) | 5 |
| 36 | call \_time | 1 | 36 | 20(%esp) | 2 |
| 37 | call \_srand | 1 |  |  |  |
| 38 | call \_rand | 1 |  |  |  |
| 39 | call \_\_Z5sort1Pdi | 1 |  |  |  |
| 40 | call \_\_Z5sort2Pdi | 1 |  |  |  |
| 41 | call \_\_Z4swapPdii | 1 |  |  |  |

В таблице 4 представлены сводные результаты расчетных характеристик вручную.

Таблица 4 – Результаты расчетных характеристик вручную

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Паскаль** | **Си** | **Ассемблер** |
| Число уникальных операторов (n1): | 17 | 24 | 41 |
| Число уникальных операндов (n2): | 18 | 18 | 36 |
| Общее число операторов (N1): | 69 | 110 | 198 |
| Общее число операндов (N2): | 88 | 95 | 326 |
| Алфавит (n): | 35 | 42 | 77 |
| Экспериментальная длина программы (Nэ): | 157 | 205 | 524 |
| Теоретическая длина программы (Nт): | 144,5455183 | 185,09775 | 405,7769322 |
| Объём программы (V): | 805,2974337 | 1105,425072 | 3283,796147 |
| Потенциальный объём (V\*): | 11,6 | 11,6 | 11,6 |
| Уровень программы (L): | 0,014404616 | 0,010493701 | 0,003532497 |
| Ожидание уровня программы (L^): | 0,024064171 | 0,015789474 | 0,005386802 |
| Интеллект программы (I): | 19,37881525 | 17,45408008 | 17,68916075 |
| Работа по программированию (Е): | 55905,51351 | 105341,7749 | 929596,3049 |
| Время кодирования (T): | 5590,551351 | 10534,17749 | 92959,63049 |
| Ожидание времени кодирования (T^): | 1409,235021 | 2943,84248 | 25372,05728 |
| Уровень языка программирования (Lambda): | 0,167093541 | 0,12172693 | 0,040976965 |
| Уровень ошибок (В): | 0,805297434 | 1,105425072 | 3,283796147 |

# Расчет метрик с помощью программы автоматизации

Для программы на Паскале:

|  |
| --- |
| Statistics for module C:\TPWDB\SOURCE\SORT.lxm  =====================================  The number of different operators : 23  The number of different operands : 21  The total number of operators : 125  The total number of operands : 87  Dictionary ( D) : 44  Length ( N) : 212  Length estimation ( ^N) : 196.281  Volume ( V) : 1157.4  Potential volume ( \*V) : 19.6515  Limit volume (\*\*V) : 38.2071  Programming level ( L) : 0.016979  Programming level estimation ( ^L) : 0.0209895  Intellect ( I) : 24.2932  Time of programming ( T) : 3787.03  Time estimation ( ^T) : 2836.29  Programming language level (lambda) : 0.333663  Work on programming ( E) : 68166.5  Error ( B) : 0.556237  Error estimation ( ^B) : 0.3858  Table:  ====================================  Operators:  | 1 | 9 | ()  | 2 | 1 | \*  | 3 | 3 | +  | 4 | 2 | -  | 5 | 45 | ;  | 6 | 13 | =  | 7 | 2 | >  | 8 | 14 | []  | 9 | 5 | ary  | 10 | 1 | boolean  | 11 | 1 | const  | 12 | 4 | for  | 13 | 2 | if  | 14 | 6 | integer  | 15 | 3 | procedure  | 16 | 1 | program  | 17 | 4 | real  | 18 | 1 | repeat  | 19 | 2 | sort1  | 20 | 2 | sort2  | 21 | 2 | swap  | 22 | 1 | type  | 23 | 1 | writeln  Operands:  | 1 | 9 | 1  | 2 | 1 | 100  | 3 | 1 | 1000  | 4 | 16 | a  | 5 | 1 | ary  | 6 | 1 | bubble\_sort  | 7 | 1 | false  | 8 | 3 | forFirstSort  | 9 | 3 | forSecondSort  | 10 | 6 | hold  | 11 | 8 | i  | 12 | 9 | j  | 13 | 5 | max  | 14 | 6 | n  | 15 | 4 | no\_change  | 16 | 3 | p  | 17 | 3 | q  | 18 | 1 | random  | 19 | 4 | randomNum  | 20 | 1 | true  | 21 | 1 | x  Summary:  =====================================  The number of different operators : 23  The number of different operands : 21  The total number of operators : 125  The total number of operands : 87  Dictionary ( D) : 44  Length ( N) : 212  Length estimation ( ^N) : 196.281  Volume ( V) : 1157.4  Potential volume ( \*V) : 19.6515  Limit volume (\*\*V) : 38.2071  Programming level ( L) : 0.016979  Programming level estimation ( ^L) : 0.0209895  Intellect ( I) : 24.2932  Time of programming ( T) : 3787.03  Time estimation ( ^T) : 2836.29  Programming language level (lambda) : 0.333663  Work on programming ( E) : 68166.5  Error ( B) : 0.556237  Error estimation ( ^B) : 0.3858 |

Для программы на Си:

(так как парсер программы на Си выдавал синтаксическую ошибки при использовании булевых переменных, все булевы переменные были заменены на целочисленные со значениями 0 и 1)

|  |
| --- |
| Statistics for module C:\TPWDB\SOURCE\SORT\_C.lxm  =====================================  The number of different operators : 33  The number of different operands : 20  The total number of operators : 182  The total number of operands : 97  Dictionary ( D) : 53  Length ( N) : 279  Length estimation ( ^N) : 252.904  Volume ( V) : 1598.09  Potential volume ( \*V) : 19.6515  Limit volume (\*\*V) : 38.2071  Programming level ( L) : 0.0122969  Programming level estimation ( ^L) : 0.0124961  Intellect ( I) : 19.9699  Time of programming ( T) : 7219.96  Time estimation ( ^T) : 6440.29  Programming language level (lambda) : 0.241652  Work on programming ( E) : 129959  Error ( B) : 0.855231  Error estimation ( ^B) : 0.532697  Table:  ====================================  Operators:  | 1 | 1 | !=  | 2 | 1 | %  | 3 | 16 | ()  | 4 | 3 | +  | 5 | 4 | ++  | 6 | 11 | ,  | 7 | 1 | -  | 8 | 40 | ;  | 9 | 4 | <  | 10 | 21 | =  | 11 | 2 | >  | 12 | 2 | SIZEOF  | 13 | 14 | []  | 14 | 1 | \_[]  | 15 | 8 | \_\_\*  | 16 | 2 | calloc  | 17 | 1 | char  | 18 | 2 | const  | 19 | 13 | double  | 20 | 1 | dowhile  | 21 | 4 | for  | 22 | 2 | if  | 23 | 13 | int  | 24 | 1 | main  | 25 | 1 | rand  | 26 | 1 | returandomNumber  | 27 | 2 | sort1  | 28 | 2 | sort2  | 29 | 1 | srand  | 30 | 2 | swap  | 31 | 1 | time  | 32 | 1 | unsigned  | 33 | 3 | void  Operands:  | 1 | 7 | 0  | 2 | 7 | 1  | 3 | 1 | 10  | 4 | 1 | 1000  | 5 | 6 | MAX  | 6 | 1 | NULL  | 7 | 16 | a  | 8 | 1 | argc  | 9 | 1 | argv  | 10 | 3 | forFirstSort  | 11 | 3 | forSecondSort  | 12 | 5 | hold  | 13 | 18 | i  | 14 | 6 | j  | 15 | 5 | n  | 16 | 4 | noChange  | 17 | 3 | p  | 18 | 3 | q  | 19 | 4 | doubleRandomNumber  | 20 | 2 | randomNumber  Summary:  =====================================  The number of different operators : 33  The number of different operands : 20  The total number of operators : 182  The total number of operands : 97  Dictionary ( D) : 53  Length ( N) : 279  Length estimation ( ^N) : 252.904  Volume ( V) : 1598.09  Potential volume ( \*V) : 19.6515  Limit volume (\*\*V) : 38.2071  Programming level ( L) : 0.0122969  Programming level estimation ( ^L) : 0.0124961  Intellect ( I) : 19.9699  Time of programming ( T) : 7219.96  Time estimation ( ^T) : 6440.29  Programming language level (lambda) : 0.241652  Work on programming ( E) : 129959  Error ( B) : 0.855231  Error estimation ( ^B) : 0.532697 |

# 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 bubble\_sort;

const max = 1000;

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

var x : ary;

i,n : integer;

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

var i,j : integer;

buffer : real;

begin

for i:=1 to n-1 do

for j:=i+1 to n do

begin

if a[i]>a[j] then

begin

buffer:=a[i];

a[i]:=a[j];

a[j]:=buffer;

end

end

end;

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

var no\_change : boolean;

j : integer;

procedure swap(var a: ary; p,q: integer);

var buffer : real;

begin

buffer:=a[p];

a[p]:=a[q];

a[q]:=buffer;

end;

begin

repeat

no\_change:=true;

for j:=1 to n-1 do

begin

if a[j]>a[j+1] then

begin

swap(a, j, j+1);

no\_change:=false;

end

end

until no\_change

end;

var firstSortedArray, secondSortedArray : ary;

doubleRandomNumber : real;

begin

randomize;

for i:=1 to max do

begin

doubleRandomNumber:=random\*1000;

forFirstSort[i]:= doubleRandomNumber;

forSecondSort[i]:= doubleRandomNumber;

end;

sort1(firstSortedArray, max);

sort2(secondSortedArray, max);

end.

# ПРИЛОЖЕНИЕ Б. ПРОГРАММА НА ЯЗЫКЕ СИ

#include <cstdlib>

#include <ctime>

const int MAX = 1000;

void firstSort(double \* a, int n)

{

double buffer;

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

if (a[j] > a[i]) {

swap(a, j, i);

}

}

}

}

void swap(double \* a, int previousPosition, int nextPosition)

{

double buffer = a[previousPosition];

a[previousPosition] = a[nextPosition];

a[nextPosition] = buffer;

}

void secondSort(double \* a, int n)

{

bool isChanged = false;

do {

isChanged = true;

for (int i = 0; i < n - 1; i++) {

if (a[i] > a[i + 1]) {

swap(a, i, i + 1);

isChanged = false;

}

}

} while (isChanged != true);

}

int main(int argc, char const \*argv[])

{

double \* firstSortedArray = (double \*)calloc(MAX, sizeof(double));

double \* secondSortedArray = (double \*)calloc(MAX, sizeof(double));

srand((unsigned)time(NULL));

double doubleRandomNumber = 0;

for (int i = 0; i < MAX; i++) {

int randomNumber = (rand() % 1000 + 1);

doubleRandomNumber = (double) randomNumber;

firstSortedArray[i] = doubleRandomNumber;

secondSortedArray[i] = doubleRandomNumber;

}

sort1(firstSortedArray, MAX);

sort2(secondSortedArray, MAX);

return randomNumber 0;

}

# ПРИЛОЖЕНИЕ В. ПРОГРАММА НА ЯЗЫКЕ АССЕМБЛЕР

.file "sort.c"

.text

.globl \_\_Z5sort1Pdi

.def \_\_Z5sort1Pdi; .scl 2; .type 32; .endef

\_\_Z5sort1Pdi:

pushl %ebp

movl %esp, %ebp

subl $16, %esp

movl $0, -4(%ebp)

jmp L2

L7:

movl $0, -8(%ebp)

jmp L3

L6:

movl -8(%ebp), %eax

leal 0(,%eax,8), %edx

movl 8(%ebp), %eax

addl %edx, %eax

fldl (%eax)

movl -4(%ebp), %eax

leal 0(,%eax,8), %edx

movl 8(%ebp), %eax

addl %edx, %eax

fldl (%eax)

fxch %st(1)

fucomip %st(1), %st

fstp %st(0)

jbe L4

L8:

movl -4(%ebp), %eax

leal 0(,%eax,8), %edx

movl 8(%ebp), %eax

addl %edx, %eax

fldl (%eax)

fstpl -16(%ebp)

movl -4(%ebp), %eax

leal 0(,%eax,8), %edx

movl 8(%ebp), %eax

addl %edx, %eax

movl -8(%ebp), %edx

leal 0(,%edx,8), %ecx

movl 8(%ebp), %edx

addl %ecx, %edx

fldl (%edx)

fstpl (%eax)

movl -8(%ebp), %eax

leal 0(,%eax,8), %edx

movl 8(%ebp), %eax

addl %edx, %eax

fldl -16(%ebp)

fstpl (%eax)

L4:

addl $1, -8(%ebp)

L3:

movl -8(%ebp), %eax

cmpl 12(%ebp), %eax

setl %al

testb %al, %al

jne L6

addl $1, -4(%ebp)

L2:

movl -4(%ebp), %eax

cmpl 12(%ebp), %eax

setl %al

testb %al, %al

jne L7

leave

ret

.globl \_\_Z4swapPdii

.def \_\_Z4swapPdii; .scl 2; .type 32; .endef

\_\_Z4swapPdii:

pushl %ebp

movl %esp, %ebp

subl $16, %esp

movl 12(%ebp), %eax

leal 0(,%eax,8), %edx

movl 8(%ebp), %eax

addl %edx, %eax

fldl (%eax)

fstpl -8(%ebp)

movl 12(%ebp), %eax

leal 0(,%eax,8), %edx

movl 8(%ebp), %eax

addl %edx, %eax

movl 16(%ebp), %edx

leal 0(,%edx,8), %ecx

movl 8(%ebp), %edx

addl %ecx, %edx

fldl (%edx)

fstpl (%eax)

movl 16(%ebp), %eax

leal 0(,%eax,8), %edx

movl 8(%ebp), %eax

addl %edx, %eax

fldl -8(%ebp)

fstpl (%eax)

leave

ret

.globl \_\_Z5sort2Pdi

.def \_\_Z5sort2Pdi; .scl 2; .type 32; .endef

\_\_Z5sort2Pdi:

pushl %ebp

movl %esp, %ebp

subl $32, %esp

movl $1, -4(%ebp)

L15:

movl $1, -4(%ebp)

movl $0, -8(%ebp)

jmp L11

L14:

movl -8(%ebp), %eax

leal 0(,%eax,8), %edx

movl 8(%ebp), %eax

addl %edx, %eax

fldl (%eax)

movl -8(%ebp), %eax

addl $1, %eax

leal 0(,%eax,8), %edx

movl 8(%ebp), %eax

addl %edx, %eax

fldl (%eax)

fxch %st(1)

fucomip %st(1), %st

fstp %st(0)

jbe L12

L16:

movl -8(%ebp), %eax

addl $1, %eax

movl %eax, 8(%esp)

movl -8(%ebp), %eax

movl %eax, 4(%esp)

movl 8(%ebp), %eax

movl %eax, (%esp)

call \_\_Z4swapPdii

movl $0, -4(%ebp)

L12:

addl $1, -8(%ebp)

L11:

movl 12(%ebp), %eax

subl $1, %eax

cmpl -8(%ebp), %eax

setg %al

testb %al, %al

jne L14

cmpl $1, -4(%ebp)

setne %al

testb %al, %al

jne L15

leave

ret

\_main:

pushl %ebp

movl %esp, %ebp

pushl %ebx

andl $-16, %esp

subl $48, %esp

call \_\_\_main

movl $8, 4(%esp)

movl $10, (%esp)

call \_calloc

movl %eax, 40(%esp)

movl $8, 4(%esp)

movl $10, (%esp)

call \_calloc

movl %eax, 36(%esp)

movl $0, (%esp)

call \_time

movl %eax, (%esp)

call \_srand

fldz

fstpl 24(%esp)

movl $0, 44(%esp)

jmp L18

L19:

call \_rand

movl %eax, %ecx

movl $274877907, %edx

movl %ecx, %eax

imull %edx

sarl $6, %edx

movl %ecx, %eax

sarl $31, %eax

movl %edx, %ebx

subl %eax, %ebx

movl %ebx, %eax

imull $1000, %eax, %eax

movl %ecx, %edx

subl %eax, %edx

movl %edx, %eax

addl $1, %eax

movl %eax, 20(%esp)

fildl 20(%esp)

fstpl 24(%esp)

movl 44(%esp), %eax

leal 0(,%eax,8), %edx

movl 40(%esp), %eax

addl %edx, %eax

fldl 24(%esp)

fstpl (%eax)

movl 44(%esp), %eax

leal 0(,%eax,8), %edx

movl 36(%esp), %eax

addl %edx, %eax

fldl 24(%esp)

fstpl (%eax)

addl $1, 44(%esp)

L18:

cmpl $9, 44(%esp)

setle %al

testb %al, %al

jne L19

movl $10, 4(%esp)

movl 40(%esp), %eax

movl %eax, (%esp)

call \_\_Z5sort1Pdi

movl $10, 4(%esp)

movl 36(%esp), %eax

movl %eax, (%esp)

call \_\_Z5sort2Pdi

movl $0, %eax

movl -4(%ebp), %ebx

leave

ret