ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ имени академика С. П. КОРОЛЕВА» (САМАРСКИЙ УНИВЕРСИТЕТ)

ИЗМЕРЕНИЕ ВРЕМЕНИ РАБОТЫ ФРАГМЕНТОВ ПРОГРАММ

Самара 2017

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА»
(САМАРСКИЙ УНИВЕРСИТЕТ)

ИЗМЕРЕНИЕ ВРЕМЕНИ РАБОТЫ ФРАГМЕНТОВ ПРОГРАММ

Составитель К.Е. Климентьев

Самара
Издательство Самарского университета
2017

УДК 004.46
ББК 32.972

Составитель ***К.Е.Климентьев***

Рецензент: к.т.н., доцент А.В. Баландин

Измерение времени работы фрагментов программ: [Электронный ресурс]: метод. указания / сост. *К.Е.Климентьев*. - Самара: Изд-во Самарского университета, 2017. – 15 с. : ил. Электрон. текстовые и граф. дан. (Кбайт).- 1 эл. опт. диск (CD-ROM)

Методические указания предназначены для студентов, изучающих в рамках направления подготовки 09.03.01 «Системы реального времени» и прочие курсы аналогичной тематики.

Содержат необходимый теоретический и справочный материал для выполнения лабораторных работ. Также могут быть использованы в курсовом проектировании и при разработке выпускной квалификационной работы.

Подготовлены на кафедре информационных систем и технологий.

УДК 004.46
ББК 32.972

Оглавление

Введение

Ц_е_л_ь л_а_б_о_р_а_т_о_р_н_о_й р_а_б_о_т_ы – ознакомление с методами измерения и расчета длительности выполнения программных фрагментов. Знание этой временно*й* характеристики важно при проектировании программно-управляемых систем реального времени.

Традиционно, для исследования свойств сложных объектов используются три группы методов:

1) методы измерения в ходе натурных экспериментов;
2) методы аналитического расчета;
3) методы имитационного моделирования.

В данной лабораторной работе методы имитационного моделирования не рассматриваются.

1. Задание на лабораторную работу

I. Написать и отладить программу, соответствующую алгоритму **A**, **B** или **C** (см. рис. 2), состоящую из фрагментов **D, E, F** или **G**.

II. Измерить длительности выполнения отдельных фрагментов при помощи способа **H** или **J**.

III. Рассчитать полную длительность *t* выполнения алгоритма, зная измеренные длительности выполнения отдельных фрагментов.

IV. Измерить полную длительность *t* выполнения алгоритма.

V. Результаты измерений и расчетов отобразить на экране (например, в окне консоли – см. рис. 1).

VI. Сравнить «измеренную» и «рассчитанную» длительности выполнения алгоритма.

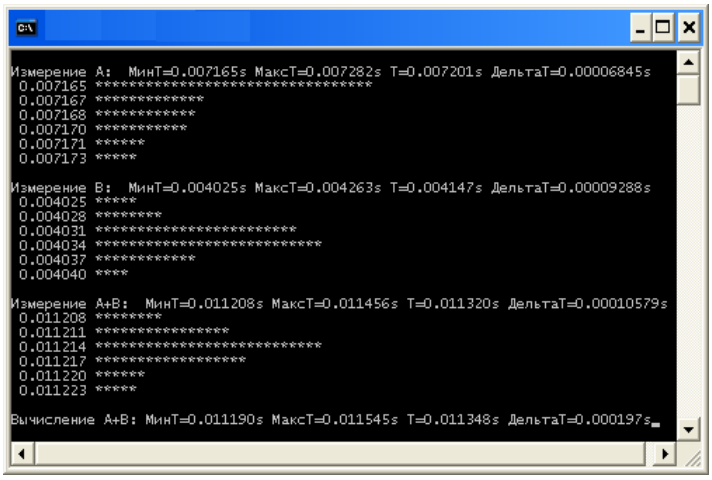
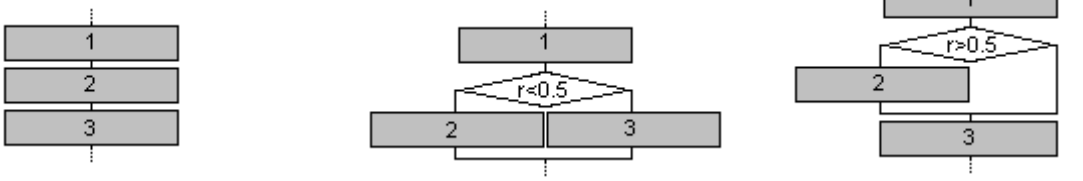


Рис. 1. Пример вывода результатов

Значения **A**, **B**, **C**, **D**, **E**, **F**, **G**, **J** и **H** для конкретного индивидуального варианта задания выбираются из Табл. 1.

Варианты схем полного алгоритма см. на рис. 2. Здесь *r*=Uniform(0,1) – непрерывная случайная величина, равномерно распределенная на интервале [0..1]. Очевидно, вместо нее можно использовать дискретную случайную величину, равновероятно принимающую значения 0 или 1.



а) Алгоритм **A** б) Алгоритм **B** в) Алгоритм **C**

Рис. 2. Варианты схем полного алгоритма

Варианты отдельных фрагментов, составляющих полный алгоритм (см. Приложение):

- **D** - Процедура сортировки методом «пузырька» массива из 1000 чисел, заполненного случайным образом.
- **E** - Процедура сортировки методом «вставок» массива из 1000 чисел, заполненного случайным образом.
- **F** - Процедура разложения на сомножители любого (на выбор) из целых чисел: 909091, 1336337, 614657 методом «brute force».
- **G** - Процедура умножения двух квадратных матриц размером 100x100, заполненных случайными значениями.

Заполнение массивов случайными значениями не является частью алгоритма, и его длительность не должна измеряться.

Варианты методов измерения времени:

- **H** - Аппаратный таймер;
- **J** - Счетчик процессорных тактов.

Таблица 1 - Индивидуальные задания

Номер варианта	Методы	Номер варианта	Методы	Номер варианта	Методы
1	ADEFH	9	CFEGH	17	BDFGH
2	BDFGJ	10	AGEDJ	18	CDEGJ
3	CDEGH	11	BGEFH	19	AEDFH
4	AEDFJ	12	CGDFJ	20	BEFGJ
5	BEFGH	13	ADEFH	21	CEDGH
6	CEDGJ	14	BDEGJ	22	AFDEJ
7	AFDEH	15	CDFGH	23	BFDGH
8	BFDGJ	16	ADEFJ	24	CFEGJ

2. Измерение временных характеристик

Для измерения временных характеристик программ используют как аппаратные измерительные мониторы, так и программные «профилировщики» (например, Intel VTune). Однако в большинстве случаев для выполнения измерения времени выполнения программных фрагментов не требуется дополнительного оборудования или программного обеспечения, кроме присутствующих в составе ЭВМ.

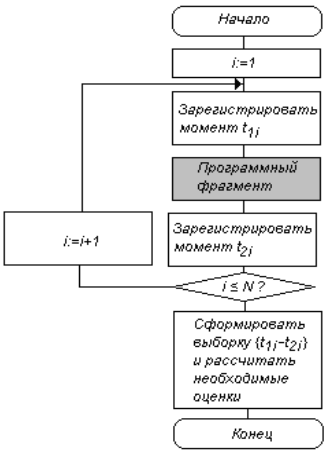


Рис. 3. Алгоритм измерений

Поскольку моделью длительности выполнения программного фрагмента является случайная величина, необходимо выполнить ее многократные измерения в цикле. Измерения производятся посредством алгоритма, изображенного на рис. 3.

2.1 Аппаратные средства измерения времени

В архитектуре персональных ЭВМ присутствуют несколько различных устройств, позволяющих отслеживать текущее время, измерять длительность временных интервалов, генерировать аппаратные прерывания в требуемые моменты и с требуемой периодичностью (см. табл. 2).

Таблица 2 – Устройства для счета времени

Устройство	Разрешающая способность
Часы реального времени (англ. RTC).	≈1 сек
Счетчик процессорных тактов (англ. TSC)	Период тактов работы процессора
Программируемый интервальный таймер (англ. PIT)	0.84 мкс
Высокоточный таймер событий (англ. HPET)	0.1 мкс и менее
Таймер монитора производительности (англ. PMTIMER)	0.28 мкс
Таймер расширенного контроллера прерываний (англ. APIC)	Зависит от материнской платы

На конкретной ЭВМ, в зависимости от года выпуска, производителя и т.п. могут присутствовать различные комбинации этих устройств. По умолчанию на современных ПЭВМ имеются (как минимум) три из них: RTC, PIT и TSC. Очевидно, для измерения коротких временных интервалов пригодны лишь PIT и TSC.

Операционные системы, как правило, используют при работе свой внутренний счетчик времени, который так же называется «таймером». Обычно такой «таймер» использует аппаратные возможности PIT-, APIC- или HPET-таймера, но имеет разрешающую способность 1 мс. Многие стандартные функции операционных систем и языков программирования (например,

GetTickCount() в Windows, currentTimeMillis() в Java, TickCount() в C# и др.) возвращают значение именного этого «таймера» и не пригодны для измерения коротких временных интервалов.

2.2. Счетчик процессорных тактов (TSC)

Микропроцессорами Intel поддерживается машинная команда RDTSC (код 310Fh), которая возвращает количество тактовых импульсов процессора, накопленное от момента включения питания. В регистр EAX помещаются младшие 32 бита счетчика, а в регистр EDX – старшие 32 бита. Каждое ядро многоядерных процессоров имеет свой собственный счетчик тактов.

Разрешающая способность метода измерения времени, использующая TSC, представляет собой величину, обратную тактовой частоте процессора.

2.2.1. Доступ к TSC в среде Windows

В Visual Studio есть встроенная функция __rdtsc(), которая сразу возвращает 64-битовое целое значение счетчика. Пример использования:

```
#include <intrin.h>
unsigned __int64 t1, t2, DeltaT;
t1 = __rdtsc();
// ... исследуемый фрагмент кода ...
t2 = __rdtsc();
DeltaT=t2-t1;
```

Если компилятор не поддерживает 64-битовые переменные или функцию _rdtsc(), то можно воспользоваться ассемблерными вставками и преобразовать два 32-битовых целых числа в одно вещественное, например:

```
float rdtsc() {
 ULONG lp, hp;
 _asm {
     rdtsc
     mov lp, eax
     mov hp, edx
  }
 return (float)lp +(float)hp*(float)0xffffffff;
}
```

2.2.2. Доступ к TSC в среде UNIX

В компиляторе GCC нет специальной функции доступа к счетчику процессорных тактов, поэтому необходимо воспользоваться встроенным ассемблером. Вот два варианта функции, ориентированные на разные версии компилятора.

```
static __inline__ unsigned long long rdtsc(void) { // Первый вариант
    unsigned long long int x;
    __asm__ volatile (".byte 0x0f, 0x31" : "=A" (x));
    return x;
```

```
}

static inline uint64_t rdtsc()  { // Второй вариант функции
  unsigned int hi, lo;
  __asm__ volatile("rdtsc" : "=a" (lo), "=d" (hi));
  return ((uint64_t)hi << 32) | lo;
}
```

2.3. Аппаратный таймер

Если на материнской плате присутствуют несколько разных таймеров (PIT, HPET, PMTIMER, APIC-таймер), то операционная система выбирает для использования наиболее «удобный». Поэтому разрешающая способность используемого системой таймера заранее не известна.

2.3.1. Доступ к аппаратному таймеру в среде Windows

Системная функция QueryPerformanceCounter с прототипом

```
BOOL QueryPerformanceCounter(
    LARGE_INTEGER  *lpliPerformanceCount // указатель на счетчик
);
```

возвращает в lpliPerformanceCount количество «тиков» таймера, накопленное с момента старта Windows. Возвращаемое значение – признак успеха (TRUE) или неудачи (FALSE).

Тип LARGE_INTEGER определяется как:

```
struct {
 DWORD LowPart; // Младшие биты 0..31
 LONG HighPart; // Старшие биты 32..62, плюс 63-й знаковый бит
};
```

Системная функция QueryPerformanceFrequency() с прототипом

```
BOOL QueryPerformanceFrequency(
    LARGE_INTEGER  *lpliPerformanceFreq // указатель на частоту
   );
```

возвращает в lpliPerformanceFreq частоту следования тактовых импульсов. Возвращаемое значение – признак успеха (TRUE) или неудачи (FALSE).

Поскольку тип LARGE_INTEGER в компиляторе определен как структура из двух 32-битовых целых чисел, то можно для выполнения арифметических действий над ними преобразовать их в одно вещественное, например:

```
#define RN(x)(float)((float)x.u.LowPart+(float)x.u.HighPart*(float)0xffffffff)
float gettimer() {
 LARGE_INTEGER f, t;
 QueryPerformanceFrequency(&f);
 QueryPerformanceCounter(&t);
 return RN(t)/RN(f);
}
```

2.3.2. Доступ к аппаратному таймеру в среде UNIX

Функция clock_gettime() возвращает количество тиков таймера, прошедших с 01.01.1970. Пример команды компиляции: gcc test.c -o ts -lrt. Описание:

```
#include < time.h >
int clock_gettime(clockid_t clock_id, struct timespec *t_time);
```

Параметры:

- clock_id – идентификатор часов (CLOCK_REALTIME – «грубые» часы на основе RTC; CLOCK_MONOTONIC – «точные» часы на основе системного таймера);
- t_time – содержит два поля: time_t tv_sec – целое количество секунд, long tv_nsec – наносекунды.

Узнать разрешающую способность часов можно с помощью функции

```
clock_getres(clockid_t clock_id, struct timespec *resolution);
```

2.4. Измерения времени в Java

Для измерения коротких временных интервалов можно использовать System.nanoTime(), но реальная разрешающая способность этого метода сильно зависит от используемого оборудования и операционной системы. Пример:

```
long t1 = System.nanoTime();
// ... исследуемый фрагмент кода ...
long t2 = System.nanoTime();
long DeltaT = t2 - t1;
```

2.5. Измерения времени в C#

Для измерений коротких временных интервалов можно использовать класс DateTime.DateTime, котоорый имеет имеет свойство Ticks, позволяющее измерять время с разрешающей способностью до "тика" длительностью 100 нс (то есть 0.0000001 с). Значение этого свойства представляет собой число 100-наносекундных интервалов, прошедших с 12:00 до полуночи, 1 января 0001 года. Пример:

```
DateTime t1 = DateTime.Now;
// ... исследуемый фрагмент кода ...
DateTime t2 = DateTime.Now;
long DeltaT = (t2.Ticks - t1.Ticks);
```

3. Обработка результатов измерений

3.1. Предварительные замечания

1. Влияние многозадачности. В Windows и разных версиях UNIX выполнение каждой задачи происходит в условиях вытесняющей многозадачности со значением кванта времени 20÷240 мс (в зависимости от версии операционной системы). Типичное распределение времени выполнения фрагментов выглядит примерно так (см. рис. 4).

Рис. 4. Плотность распределения времени

На этом рисунке «истинное» время выполнения фрагмента программы соответствует левой части гистограммы, а правую часть составляют задержки, вызванные прерываниями выполнения программы со стороны других. Таким образом, необходимо производить расчеты только по значениям, меньшим, чем некоторое «пороговое» значение.

2. Влияние кэширования и предварительной компиляции. При циклическом выполнении программных фрагментов процессор при первой итерации загружает команды в свой быстродействующий буфер (кэш), а во время следующих операций обращения к оперативной памяти не происходит. Следовательно, продолжительность выполнения самой первой итерации будет заметно больше, чем других.

Такой же эффект возникает и при использовании языков типа Java или C#, но обусловлен тем, что при первой итерации команды загружаются в виртуальную машину и предварительно компилируются.

При расчетах необходимо исключить из выборки самый первый отсчет.

3.2. Расчет статистических характеристик

Методика обработки результатов *N*-кратных измерений времени *t* использует рекомендации ГОСТ 8.207-76 и СТО СГАУ 02068410-009-2007.

1. За оценку значения *t* принимают среднее арифметическое множества $\{t_i\}$ результатов отдельных наблюдений:

$$\tilde{t} = \frac{1}{N}\sum_{i=1}^{N} t_i \,.$$

2. По множеству $\{\tilde{t} - t_i\}$ (или просто по $\{t_i\}$) определяется закон распределения. В ГОСТ 8.207-76 описаны формальные методы, но в рамках лабораторной работы достаточно построить гистограмму частот и оценить закон распределения «на глаз». При этом количество *K* дифференциальных

классов («столбиков») гистограммы можно выбрать в соответствии с правилом Стургесса: $K = \lceil log_2 N \rceil + 1$, где [.] – операция округления.

3. Если распределение близко к нормальному, то по $\{t_i\}$ сначала вычисляется оценка с.к.о. случайной погрешности результата измерения:

$$\tilde{\sigma}_t = \sqrt{\frac{1}{N(N-1)} \sum_{i=1}^{N} \left(t_i - \tilde{t}\right)^2} = \sqrt{\left(\frac{1}{N} \sum_{i=1}^{N} t_i^2 - \tilde{t}^2 \right) / N} .$$

Пределы $\pm\Delta t$ доверительного интервала, в котором находится истинное значение t, рассчитываются по формуле $\Delta t = \gamma \times \tilde{\sigma}_t$, где γ - коэффициент Стьюдента (в случае объема выборки $N<29$) или квантиль нормального распределения (в случае объема выборки $N \geq 29$), который выбирается из табл. 3 в соответствии с желаемой доверительной вероятностью (например, $P_д=0.95$).

Таблица 3 - Квантили нормального распределения

$P_д$	0.9	0.95	0.99	0.997	0.999
γ	1.60	1.96	2.58	3.00	3.29

Если распределение отлично от нормального, то по множеству $\{t_i\}$ рассчитываются максимальное t_{max} и минимальное t_{min} значения, и в соответствии с РТМ 25 139-74 принимают $\Delta t = max\left(t_{max} - \tilde{t}, \tilde{t} - t_{min}\right)$. При этом необходимо иметь в виду, что доверительная вероятность $P_д=0.95$ оценки достигается при $N \geq 29$.

4. Результат измерений предоставляется в виде: $\tilde{t} \pm \Delta t$. При необходимости так же указывается $P_д$.

3.3. Определение объема выборки

Как абсолютная точность ε (или относительная точность ε_0), так и достоверность $P_д$ результата измерения зависят от объема выборки N, используемой для получения оценок.

Если в результате измерений оценивается вероятность p' наступления какого-либо события (например, невыхода модуля случайной величины за наперед заданные пределы), то принимают:

$$N \geq \gamma^2 p'(1-p')/\varepsilon \ = \gamma^2 (1-p')/\left(\varepsilon_0^2 p'\right).$$

Если оценивается математическое ожидание m', то принимают:

$$N \geq \gamma^2 \sigma'^2 / \varepsilon^2 = \gamma^2 \sigma'^2 / \left(\varepsilon_0^2 m'\right).$$

Здесь квантиль γ выбирается из табл. 3. Обычно в технических измерениях используют значения $\varepsilon_0=0.01$ и $P_д=0.95$. Кроме того, поскольку p', m' и σ' заранее не известны, то в формулах используются их грубые оценки, рассчитанные по малой части выборки.

Кроме того, эти формулы позволяют решать обратные задачи, например, определять ε_0 по известному N.

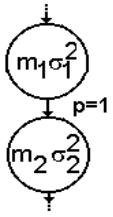
4. Методы аналитического расчета

Эта большая группа методов подразумевает расчет необходимых характеристик системы на основе ее аналитической модели. Исходные числовые параметры для этих аналитических моделей обычно получаются методами натурных измерений.

Идея основана на представлении программы в виде ориентированного графа, вершинам которого ставятся в соответствие программные блоки, а дугам - переходы между блоками. Вершины графа маркируются значениями статистических характеристик времени исполнения, а дуги - вероятностями переходов.

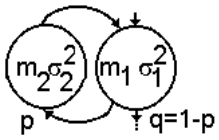
Построенный таким образом граф подвергается «редукции», т.е. замене групп вершин отдельными вершинами с обобщенными значениями математического ожидания m и дисперсии σ^2 по следующим правилам.

1. Устранение последовательности.



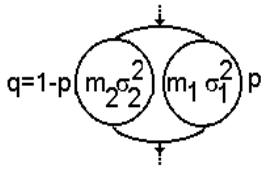
$$m = m_1 + m_2$$
$$\sigma^2 = \sigma^2_1 + \sigma^2_2$$
$$t_{max} = t_{max1} + t_{max2}$$
$$t_{min} = t_{min1} + t_{min2}$$

2. Устранение петли.



$$m = m_1 + (m_1 + m_2)\times (p/1\text{-}p)$$
$$\sigma^2 = \sigma^2_1 + (\sigma^2_2 + \sigma^2_2) \times (p/1\text{-}p)$$
$$t_{min} = t_{min1}$$
$$t_{max} = t_{max1} + (t_{max1} + t_{max2}) \times (p/1\text{-}p)$$

3. Устранение развилки.



$$m = m_1\times p + m_2\times(1\text{-}p)$$
$$\sigma^2 = \sigma^2_1\times p + \sigma^2_2\times (1\text{-}p)$$
$$t_{max} = \max(t_{max1}, t_{max2})$$
$$t_{min} = \min(t_{min1}, t_{min2})$$

Итогом редукции является формула, описывающая математическое ожидание m и дисперсию σ^2 времени выполнения фрагмента программы через математическое ожидание и дисперсии его составных частей.

Расчет предельных значений Δt_{min} и Δt_{max} по рассчитанным m и σ^2 для нормально распределенной измеряемой величины рассмотрен в п. 3.2. В случае, если распределение отлично от нормального, можно воспользоваться «сложением» Δt_{min} и Δt_{max} по правилам интервальной арифметики (см. выше).

Литература

1. Советов, Яковлев. Моделирование систем. – М.: Высшая школа, 2001. – 343 с.

2. ГОСТ 8.207-76. Прямые измерения с многократными наблюдениями. Методы обработки результатов наблюдений. – М.: Стандартинформ, 2006. – 7 с.

3. СТО СГАУ 02068410-009-2007. Обработка и оформление результатов измерений. – Самара: изд-во СГАУ, 2007. - 34 с.

4. РТМ 25 139-74. Методы нормирования метрологических характеристик, оценки и контроля характеристик погрешностей средств статистических измерений. – М.: Минприбор, 1974. – 76 с.

5. Гайдышев И. Анализ и обработка данных. Специальный справочник. – СПб: Питер, 2001. – 752 с.

Приложение. Алгоритмы отдельных фрагментов на псевдокоде

// Сортировка вставками массива a
`for i ∈ [1, n-1]`
`  j ← i-1`
`  while j ≠ 0 & a[j]>a[j+1]`
`  a[j]↔a[j + 1]`
`  j ← j-1`
	// Сортировка пузырьком массива a
	`for i ∈ [0,n-2]`
	`  for j ∈ [0, n-i-2]`
	`  if (a[j] > a[j + 1])`
	`  a[j]↔a[j + 1]`

// Умножение матриц C = A*B
`for i ∈ [0, n-1]`
`  for j ∈ [0, n-1]`
`  C[i][j] ← 0;`
`  for k ∈ [0, n-1]`
`  C[i][j] ← C[i][j]+A[i][k]*B[k][j];`
	// Определение простоты числа n
	`i ← 2;`
	`while (i<`$\sqrt{n}$`) & (n mod i ≠ 0)`
	`  i ← i+1;`
	`  result ← (i=`$\sqrt{n}$`)`

Методические материалы

ИЗМЕРЕНИЕ ВРЕМЕНИ РАБОТЫ ФРАГМЕНТОВ ПРОГРАММ

Методические указания

Составитель ***Климентьев Константин Евгеньевич***

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ имени академика С.П.Королева»
(Самарский университет)
443086, САМАРА, МОСКОВСКОЕ ШОССЕ, 34

Изд-во Самарского университета
443086, Самара, Московское шоссе, 34.