

**Звіт з лабораторної роботи
за дисципліною "Архітектура і програмне
забезпечення обчислювальних систем"
студента групи ПА-17-1
Панасенка Егора Сергійовича
Кафедра комп'ютерних технологій, фпм, дну
2017/2018 навч.р.**

1. Постановка задачі: Зібрати статистику швидкості різних типів множення матриці на вектор з різними типами даних. Такі типи множення: множення на вектор з переходами по стовпцям (Метод 1), множення на вектор з переходами по строкам (Метод 2), множення строки на матрицю (Метод 3). Використати такі типи даних: ціле (int), збільшене ціле (long long int) та дробове (double).
2. Опис роботи коду:
 1. big_collector.sh
 1. Запускає collector.sh для кожної з 9 програм.
 2. collector.sh
 1. Створює заново csv файл статистики
 2. Компілює потрібну програму
 3. Перевіряє скільки доступно пам'яті.
 4. Виконує цикл доки не закінчиться пам'ять
 1. Запускає відладчик для того щоб забрати кількість використаної пам'яті
 2. Перевіряє скільки доступно пам'яті після виконання.
 5. 01.c-03.c – програми мають тільки різний тип даних (int, long long int, long double)
 1. Ініціювання потрібних змінних
 2. Зберігання початкового часу
 3. Ініціювання матриці з векторами
 4. Заповнювання випадковими числами від -20 до 20 матриці та вектора
 5. Множення матриці на вектор з переходом по стовпцям, тобто беремо строку матриці множимо її елементи на відповідні елементи вектора та додаємо отримані відповіді і присвоюємо це значення до елемента з номером

строки матриці вихідного вектора. (Метод 1)

6. Виведення часу виконання

7. Звільнення пам'яті

8. Вихід програми

6. 04.c-07.c – програми мають тільки різний тип даних (int, long long int, long double)

1. Ініціювання потрібних змінних

2. Зберігання початкового часу

3. Ініціювання матриці з векторами

4. Заповнювання випадковими числами від -20 до 20 матриці та вектора

5. Множення матриці на вектор з переходом по строкам, тобто беремо стовпець матриці множимо її елементи на елемент вектора з номером стовпця та додаємо це значення до елемента з номером строки матриці вихідного вектора. (Метод 2)

6. Виведення часу виконання

7. Звільнення пам'яті

8. Вихід програми

7. 04.c-07.c – програми мають тільки різний тип даних (int, long long int, long double)

1. Ініціювання потрібних змінних

2. Зберігання початкового часу

3. Ініціювання матриці з векторами

4. Заповнювання випадковими числами від -20 до 20 матриці та вектора

5. Стандартне множення вектора (строку) на матрицю (Метод 3)

6. Виведення часу виконання

7. Звільнення пам'яті

8. Вихід програми

3. Опис результатів:

1. Результати показали, що 2 метод працює дуже повільно (у 2 рази повільніше), а 1 та 3 однаково швидко, хоча усі вони відрізняються. Пояснюється це тим, що при читанні масиву, у кеш пам'ять переходить не одна частина масиву, а увесь масив,

а якщо масив дуже великий, то береться його частина. У 1 методі ми брали підряд елементи рядка матриці і елементи вектора, а у 2 методі не брали жоден елемент підряд, тому знижується швидкість, так як програмі треба брати код не з кеша, а безпосередньо з пам'яті, то знижується швидкість. У 3 методі ми брали підряд елементи вихідного вектора і елементи рядка матриці. Також дуже цікаво описано про кеш пам'ять тут <https://stackoverflow.com/questions/16699247/what-is-cache-friendly-code>

4. Інформація про комп'ютер:

CPU GPU Mobo Audio Drives System

INTEL

(R) Core(TM) i3-3217U CPU
1795.928 Mhz

TEMP: 56.0°C CRIT: 105.0°C

amily: 6 | Model: 58 | Stepping: 9 | Cores: 4 | Phy id: 0
Address sizes: 36 bits physical, 48 bits virtual

Flags: MMX, SSE, SSE2, XD-Bit, SSE3, SSE3.1, SSE4.1, SSE4.2, **More...**

x86 64-Bit Ext: ☒ HW Virt: ☒ Hyper-Threading: ☒

Power mangament:

Data from the manufacturer |..... Bogomips: 3591.85

Integrated GPU: Intel HD 4000 Codename: Ivy Bridge

TDP: 17 W Package: FC-BGA12F

PPS: 22 nm TCaseMax: unknown

Socket: Intel BGA1023 Turboclock: NA

Transistors: unknown Multiplier: 18.0x

Die size: 118 mm? Voltage: unknown

Cache L1 32K Data WOA: 8 CLS: 64 NOS: 64

Cache L1 32K Instruction WOA: 8 CLS: 64 NOS: 64

Cache L2 256K Unified WOA: 8 CLS: 64 NOS: 512

Cache L3 3072K Unified WOA: 12 CLS: 64 NOS: 4096

CPUINFO CPUID FLAG: 5%

7.6.0 cpu3 Close

CPU GPU Mobo Audio Drives System

CPU Vendor: GenuineIntel
CPU Codename: Sandy Bridge (Core i3)
CPU Brand: Intel(R) Core(TM) i3-3217U CPU @ 1.80GHz
CPU Vendor: Intel CPU
CPU Family: 6
CPU Model: 10
CPU Stepping: 9
CPU Ext family: 6
CPU Ext model: 58
CPU Num cores: 2
CPU Num logical cpus: 4
CPU Total logical cpus: 4
L1 Data Cache: 32 KB, Assoc: 8-way, Cacheline: 64 byte
L1 Instruction Cache: 32 KB
L2 Cache: 256 KB, Assoc: 8-way, Cacheline: 64 byte
L3 Cache: 3072 KB, Assoc: 12-way, Cacheline: 64 byte
L4 Cache: -1 KB, Assoc: undetermined, Cacheline: undetermined
SSE Size: 128 bit (non-authoritative)
Clock: 1795 Mhz, By OS: 1795 Mhz, By IC: 1773 Mhz, Measure: 1795 Mhz
CPU Mark TSC: 5104, Sys Clock: 2

CPUINFO CPUID FLAG: 2%

7.6.0 Close

CPU GPU Mobo Audio Drives System

Intel Corporation 3rd Gen Core processor Graphics Controller

OpenGL Vendor: Intel Open Source Technology Center
OpenGL Renderer: Mesa DRI Intel(R) Ivybridge Mobile
OpenGL Version: 3.0 Mesa 17.2.8
Client glx vendor: Mesa Project and SGI
Client glx version: 1.4
Server glx vendor: SGI
Server glx version: 1.4
GLX version: 1.4
OpenGL PVS: OpenGL ES 3.0 Mesa 17.2.8
OpenGL PSLVS: OpenGL ES GLSL ES 3.00

Kernel driver in use: i915
Total Memory prefetchable: 4M -
Total Memory non-prefetchable: 256M -
Present resolution: 1366x768 pixels (361x203 millimeters)
minimum 8 x 8, current 1366 x 768, maximum 32767 x 32767

EDID

Manufacturer: AUO Model 47ec Serial Number 0
Monitor name: AUO B156XW00DPMS: On Gamma: 2.20
Status: connected Enabled: enabled
Maximum image size: 34 cm x 19 cm Serial number:

7.6.0 Force card0-eDP: Close

CPU GPU Mobo Audio Drives System

Board

Board Vendor: Acer
Board Version: V2.06
Board Name: EA50_CX
Board Asset Tag: Type 2 - Board Asset Tag

Bios

Bios Vendor: Insyde Corp.
Bios Version: V2.06
Bios Date: 10/08/2013

Chassis

Chassis Vendor: Insyde Corp.
Chassis Version: V2.06
Chassis Type: Acer Asset Tag String
Chassis Asset Tag: Aspire E1-570G

Product

Product Name: Aspire E1-570G
Product Version: V2.06

7.6.0 Close

CPU GPU Mobo Audio Drives System

HDA-Intel - HDA Intel PCH input10

/proc/asound/PCH/codect#0.Codect: Realtek HDA Intel PCH Headphone

ALSA Driver Version k4.13.0-37-generic.

Node 0x20 [Vendor Defined Widget] wcaps 0xf00040: Mono
Processing caps: benign=0, ncoeff=117
Node 0x21 [Pin Complex] wcaps 0x40058d: Stereo Amp-Out
Control: name="Headphone Playback Switch", index=0, device=0
ControlAmp: chs=3, dir=Out, idx=0, ofs=0
Amp-Out caps: ofs=0x00, nsteps=0x00, stepsize=0x00, mute=1
Amp-Out vals: {0x00 0x00}
Pinconf 0x0000001c: OUT HP Detect
Pin Default 0x0321101f: [Jack] HP Out at Ext Left
Conn = 116, Color = Black
DefAssociation = 0x1, Sequence = 0xf
Pin-ctls: 0xc0: OUT HP
Unsolicited: tag=01, enabled=1
Power: states: D0 D1 D2 D3 EPSS
Power: setting=D3, actual=D3
Connection: 2
0x0c 0x0d*
Node 0x22 [Audio Mixer] wcaps 0x20010b: Stereo Amp-In
Amp-In caps: ofs=0x00, nsteps=0x00, stepsize=0x00, mute=1
Amp-In vals: {0x80 0x80} [0x80 0x80] [0x80 0x80] [0x80 0x80] [0x
Connection: 6
0x18 0x19 0x1a 0x1b 0x1d 0xb0
Node 0x23 [Audio Mixer] wcaps 0x20010b: Stereo Amp-In
Amp-In caps: ofs=0x00, nsteps=0x00, stepsize=0x00, mute=1
Amp-In vals: {0x80 0x80} [0x80 0x80] [0x80 0x80] [0x00 0x00] [0x80 0x80] [0x
Connection: 6
0x18 0x19 0x1a 0x1b 0x1d 0xb0

7.6.0 Close

CPU GPU Mobo Audio Drives System

500.1 GB Vendor: ATA Device rev: SDM1

Model: ST500LT012-9WS14 Filesystem: SWAP

State: running Rotational: 1 Removable: No Port: SATA0

Point:

UUID: 2ade8b72-2cd2-4ad1-90b6-94b12750e8d7

Width:

0%

Disk size: 465.7 GB Part size: 954 MB Free: 954

Modallias: scsi-l-0x00 HW SATA SPD Limit: 6.0 Gbps
Add random: 1 SATA SPD: 3.0 Gbps
Physical block size: 4096 SATA SPD Limit: 6.0 Gbps
Logical block size: 512 Range: 16 Ext range: 256
Minimum IO size: 4096 Max sectors: Max sectors kb: 1280
Optimal IO size: 0 Max HW sectors kb: 32767
HW sector size: 512 Max segments: 128 Max segment size: 65536
Rq affinity: 1 Discard max bytes: 0 Discard granularity: 0
IO stats: 1 Discard zeroes data: 0
Read ahead kb: 128 BDI stable pages required: 0
Nr requests: 128 BDI min_ratio: 0
Nomerges: 0 BDI max_ratio: 100

7.6.0 sda sda4 Close

CPU GPU Mobo Audio Drives System

Distro Vendor: Ubuntu
Distro Release: 16.04
Desktop environment: LXDE
Window manager: OpenBox
Window manager Theme: Lubuntu-small
GTK+ 2 Theme: Lubuntu-default
GTK+ 3 Theme: Not Found
GTK Font: Comic Jens Free Pro 8
GTK Icons: Lubuntu
System SHELL: /bin/sh

GCC Version: 5.4.0 Linux C Library:
Hostname: gaura GNU Make Version: 4.1
Arch: x86_64 PPP: 2.4.7
Timezone: Europe/Zaporozhye Dynamic linker (ldd): 2.23
X.Org Version: 1.19.5 Net-tools: 1.60
GLX Version: 1.4 Wireless-tools: 30
Binutils 2.26.1
default Display Manager: /usr/sbin/lightdm

Uptime: 03:10:48

7.6.0 Close

Mobo Audio Drives System Kernel Memory

Machine: x86_64
OS Type: Linux
Hostname: gaura
OS Release: 4.13.0-37-generic
Build Date: #42~16.04.1-Ubuntu SMP Wed Mar 7 16:03:28 UTC 2018

Installed kernels: 2

vmlinuz-4.13.0-36-generic
vmlinuz-4.13.0-37-generic

BOOT IMAGE=/boot/vmlinuz-4.13.0-37-generic root=UUID=bce79534-7b8a-45bc-b12e-96407068cda ro quiet splash rdblacklist=nouveau resume=/dev/disk/by-uuid/2ade8b72-2cd2-4ad1-90b6-94b12750e8d7 vt.handoff=7

7.6.0 Show modules Close

Mobo Audio Drives System Kernel Memory

Standard Info Serial Presence Detect (Beta)

Mem used: 1493.332 MB
38%

Swap used: 0 MB
0%

Heap size is 6,160,384 bytes.

Mem Total: 3929.476MB
Mem Free: 520.908MB
Mem Available: 1843.056MB
Buffers: 117.828MB
Cached: 1797.408MB
Swap Free: 976.892MB Swap Cached: 0MB
Swap Total: 976.892MB Active: 2136.808MB

7.6.0 Close

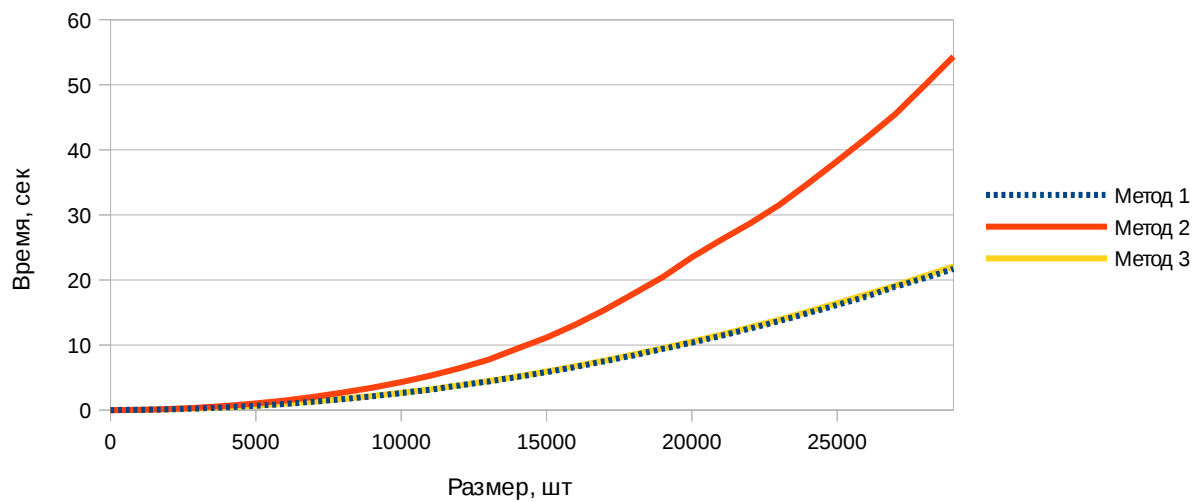
5. Таблиці та графіки:

| Построчное умножение матрицы на вектор | | | | | | | | |
|--|-------|--------|---------------|-------|--------|-------------|------|--------|
| int | | | long long int | | | long double | | |
| Size | Time | Memory | Size | Time | Memory | Size | Time | Memory |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1000 | 0.03 | 3 | 1000 | 0.04 | 7 | 1000 | 0.05 | 15 |
| 2000 | 0.11 | 15 | 2000 | 0.13 | 30 | 2000 | 0.17 | 61 |
| 3000 | 0.24 | 34 | 3000 | 0.26 | 68 | 3000 | 0.36 | 137 |
| 4000 | 0.42 | 61 | 4000 | 0.46 | 122 | 4000 | 0.63 | 244 |
| 5000 | 0.66 | 95 | 5000 | 0.71 | 190 | 5000 | 0.98 | 381 |
| 6000 | 0.94 | 137 | 6000 | 1.02 | 274 | 6000 | 1.4 | 549 |
| 7000 | 1.28 | 187 | 7000 | 1.38 | 374 | 7000 | 1.9 | 748 |
| 8000 | 1.67 | 244 | 8000 | 1.79 | 488 | 8000 | 2.46 | 977 |
| 9000 | 2.11 | 309 | 9000 | 2.28 | 618 | 9000 | 3.12 | 1266 |
| 10000 | 2.6 | 381 | 10000 | 2.82 | 763 | 10000 | 3.84 | 1563 |
| 11000 | 3.14 | 462 | 11000 | 3.39 | 923 | 11000 | 4.62 | 1848 |
| 12000 | 3.77 | 549 | 12000 | 4.02 | 1099 | 12000 | 5.51 | 2203 |
| 13000 | 4.38 | 645 | 13000 | 4.75 | 1289 | 13000 | 6.47 | 2590 |
| 14000 | 5.1 | 748 | 14000 | 5.46 | 1496 | 14000 | 7.53 | 3008 |
| 15000 | 5.83 | 858 | 15000 | 6.25 | 1717 | 15000 | 8.61 | 3457 |
| 16000 | 6.62 | 977 | 16000 | 7.11 | 1953 | | | |
| 17000 | 7.51 | 1103 | 17000 | 8.62 | 2258 | | | |
| 18000 | 8.41 | 1236 | 18000 | 9.64 | 2531 | | | |
| 19000 | 9.4 | 1377 | 19000 | 10.74 | 2820 | | | |
| 20000 | 10.36 | 1526 | 20000 | 11.89 | 3125 | | | |
| 21000 | 11.41 | 1682 | 21000 | 13.12 | 3445 | | | |
| 22000 | 12.52 | 1847 | | | | | | |
| 23000 | 13.68 | 2018 | | | | | | |
| 24000 | 14.91 | 2198 | | | | | | |
| 25000 | 16.16 | 2384 | | | | | | |
| 26000 | 17.46 | 2579 | | | | | | |
| 27000 | 18.97 | 2781 | | | | | | |
| 28000 | 20.27 | 2991 | | | | | | |
| 29000 | 21.7 | 3209 | | | | | | |

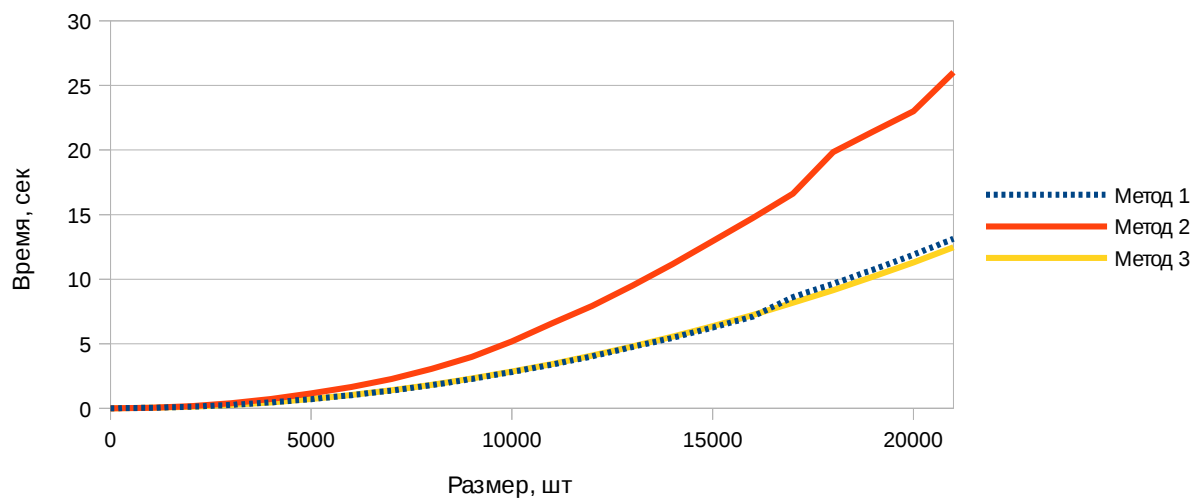
| Постолбцовое умножение матрицы на вектор | | | | | | | | |
|--|-------|--------|---------------|-------|--------|-------------|-------|--------|
| int | | | long long int | | | long double | | |
| Size | Time | Memory | Size | Time | Memory | Size | Time | Memory |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1000 | 0.05 | 3 | 1000 | 0.05 | 7 | 1000 | 0.07 | 15 |
| 2000 | 0.15 | 15 | 2000 | 0.17 | 30 | 2000 | 0.26 | 61 |
| 3000 | 0.35 | 34 | 3000 | 0.4 | 68 | 3000 | 0.58 | 137 |
| 4000 | 0.63 | 61 | 4000 | 0.73 | 122 | 4000 | 1.01 | 244 |
| 5000 | 1.01 | 95 | 5000 | 1.15 | 190 | 5000 | 1.6 | 381 |
| 6000 | 1.46 | 137 | 6000 | 1.65 | 274 | 6000 | 2.35 | 549 |
| 7000 | 2.02 | 187 | 7000 | 2.27 | 374 | 7000 | 3.17 | 748 |
| 8000 | 2.7 | 244 | 8000 | 3.05 | 488 | 8000 | 4.08 | 977 |
| 9000 | 3.44 | 309 | 9000 | 3.99 | 618 | 9000 | 5.46 | 1266 |
| 10000 | 4.3 | 381 | 10000 | 5.18 | 763 | 10000 | 6.55 | 1563 |
| 11000 | 5.27 | 462 | 11000 | 6.58 | 923 | 11000 | 8.24 | 1848 |
| 12000 | 6.42 | 549 | 12000 | 7.92 | 1099 | 12000 | 9.67 | 2203 |
| 13000 | 7.75 | 645 | 13000 | 9.5 | 1289 | 13000 | 11.25 | 2590 |
| 14000 | 9.47 | 748 | 14000 | 11.17 | 1496 | 14000 | 13.37 | 3008 |
| 15000 | 11.12 | 858 | 15000 | 12.95 | 1717 | 15000 | 14.68 | 3457 |
| 16000 | 13.13 | 977 | 16000 | 14.74 | 1953 | | | |
| 17000 | 15.42 | 1103 | 17000 | 16.61 | 2258 | | | |
| 18000 | 17.9 | 1236 | 18000 | 19.84 | 2531 | | | |
| 19000 | 20.45 | 1377 | 19000 | 21.44 | 2820 | | | |
| 20000 | 23.48 | 1526 | 20000 | 22.99 | 3125 | | | |
| 21000 | 26.15 | 1682 | 21000 | 26.02 | 3445 | | | |
| 22000 | 28.68 | 1847 | | | | | | |
| 23000 | 31.53 | 2018 | | | | | | |
| 24000 | 34.84 | 2198 | | | | | | |
| 25000 | 38.27 | 2384 | | | | | | |
| 26000 | 41.82 | 2579 | | | | | | |
| 27000 | 45.48 | 2781 | | | | | | |
| 28000 | 49.87 | 2991 | | | | | | |
| 29000 | 54.36 | 3209 | | | | | | |

| Умножение строки на матрицу | | | | | | | | |
|-----------------------------|-------|--------|---------------|-------|--------|-------------|------|--------|
| int | | | long long int | | | long double | | |
| Size | Time | Memory | Size | Time | Memory | Size | Time | Memory |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1000 | 0.03 | 3 | 1000 | 0.04 | 7 | 1000 | 0.05 | 15 |
| 2000 | 0.11 | 15 | 2000 | 0.12 | 30 | 2000 | 0.17 | 61 |
| 3000 | 0.25 | 34 | 3000 | 0.27 | 68 | 3000 | 0.36 | 137 |
| 4000 | 0.43 | 61 | 4000 | 0.47 | 122 | 4000 | 0.63 | 244 |
| 5000 | 0.67 | 95 | 5000 | 0.72 | 190 | 5000 | 0.98 | 381 |
| 6000 | 0.96 | 137 | 6000 | 1.04 | 274 | 6000 | 1.42 | 549 |
| 7000 | 1.3 | 187 | 7000 | 1.4 | 374 | 7000 | 1.89 | 748 |
| 8000 | 1.69 | 244 | 8000 | 1.82 | 488 | 8000 | 2.48 | 977 |
| 9000 | 2.15 | 309 | 9000 | 2.31 | 618 | 9000 | 3.12 | 1266 |
| 10000 | 2.64 | 381 | 10000 | 2.85 | 763 | 10000 | 3.86 | 1563 |
| 11000 | 3.2 | 462 | 11000 | 3.44 | 923 | 11000 | 4.64 | 1848 |
| 12000 | 3.81 | 549 | 12000 | 4.09 | 1099 | 12000 | 5.49 | 2203 |
| 13000 | 4.46 | 645 | 13000 | 4.79 | 1289 | 13000 | 6.5 | 2590 |
| 14000 | 5.17 | 748 | 14000 | 5.54 | 1496 | 14000 | 7.48 | 3008 |
| 15000 | 5.92 | 858 | 15000 | 6.35 | 1717 | 15000 | 8.62 | 3457 |
| 16000 | 6.73 | 977 | 16000 | 7.23 | 1953 | | | |
| 17000 | 7.62 | 1103 | 17000 | 8.18 | 2258 | | | |
| 18000 | 8.54 | 1236 | 18000 | 9.15 | 2531 | | | |
| 19000 | 9.5 | 1377 | 19000 | 10.21 | 2820 | | | |
| 20000 | 10.53 | 1526 | 20000 | 11.29 | 3125 | | | |
| 21000 | 11.61 | 1682 | 21000 | 12.48 | 3445 | | | |
| 22000 | 12.75 | 1847 | | | | | | |
| 23000 | 13.9 | 2018 | | | | | | |
| 24000 | 15.13 | 2198 | | | | | | |
| 25000 | 16.41 | 2384 | | | | | | |
| 26000 | 17.76 | 2579 | | | | | | |
| 27000 | 19.14 | 2781 | | | | | | |
| 28000 | 20.59 | 2991 | | | | | | |
| 29000 | 22.07 | 3209 | | | | | | |

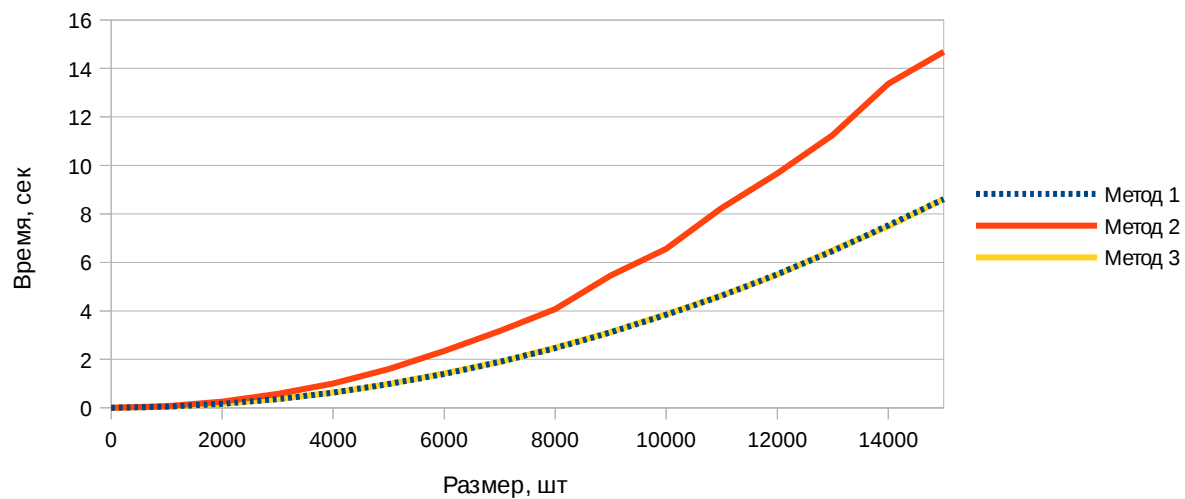
Integer



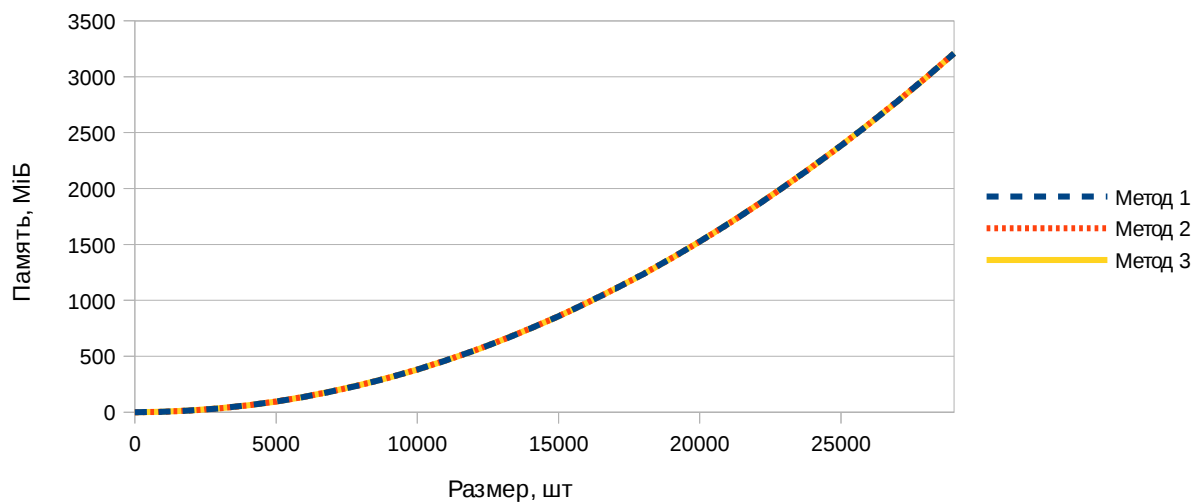
Long long integer



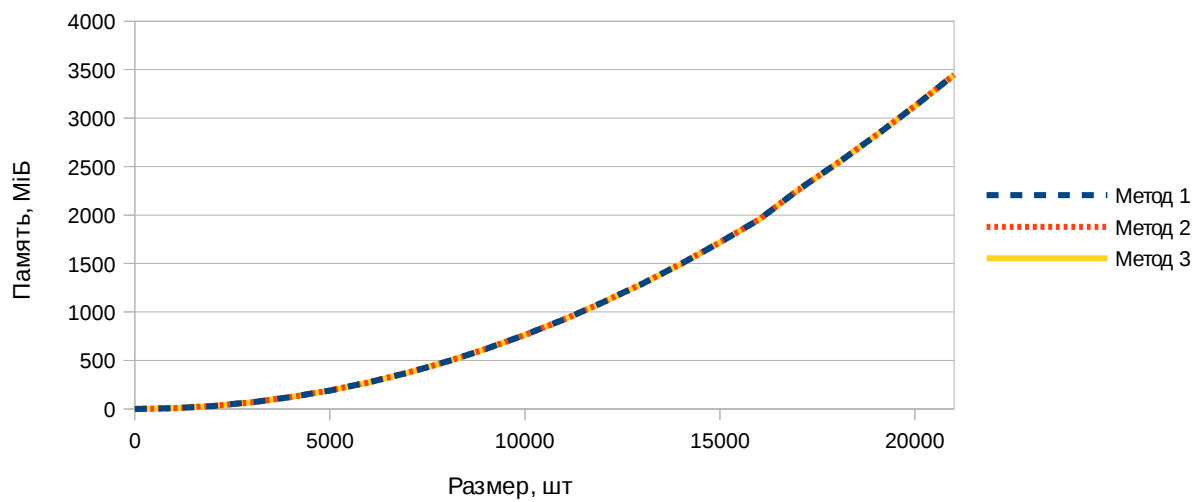
Long double



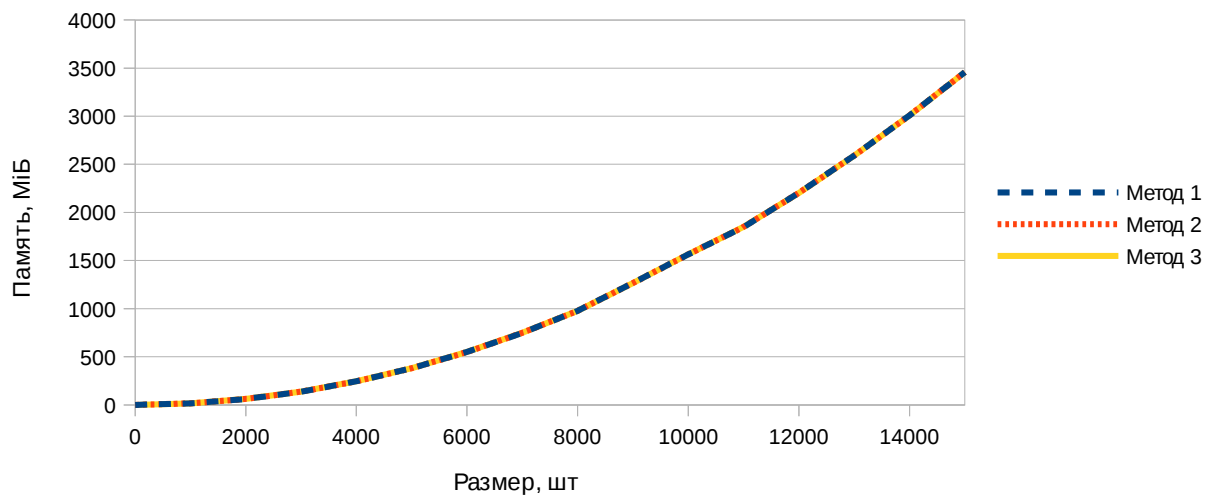
Integer



Long long integer



Long double



6. Коди програм, скриптів:

big_collector.sh

```
#!/bin/bash

./collector.sh 01.c 01.csv 0 1000
./collector.sh 02.c 02.csv 0 1000
./collector.sh 03.c 03.csv 0 1000
./collector.sh 04.c 04.csv 0 1000
./collector.sh 05.c 05.csv 0 1000
./collector.sh 06.c 06.csv 0 1000
./collector.sh 07.c 07.csv 0 1000
./collector.sh 08.c 08.csv 0 1000
./collector.sh 09.c 09.csv 0 1000
exit 0
```

collector.sh

```
#!/bin/bash

SOURCE_CODE="main.c"
if [ ! -z "$1" ]
then
    SOURCE_CODE=$1
fi
OUTPUT_FILE="statistics.txt"
if [ ! -z "$2" ]
then
    OUTPUT_FILE=$2
fi
BEGIN="1000"
if [ ! -z "$3" ]
then
    BEGIN=$3
fi
STEP="1000"
if [ ! -z "$4" ]
then
    STEP=$4
fi

rm $OUTPUT_FILE
echo "Size,Time,Memory" > $OUTPUT_FILE
gcc -g $SOURCE_CODE
read -r -a MEM <<< "$(free -m | tr -s " " | grep Mem)"
FREE_MEM=`echo "(${MEM[1]}*85/100-${MEM[2]}-${MEM[4]})" | bc`
USED_MEMORY=0
i=$BEGIN
while [ $USED_MEMORY -le $FREE_MEM ]
do
    USED_MEMORY=`echo "break 25"
run $i >> $OUTPUT_FILE
call malloc_stats()
continue
```

```

quit" | gdb a.out 2>&1 | grep "system bytes" | head -2 | tail -1 | tr -s " " |
cut -d " " -f 4`
echo $USED_MEMORY
USED_MEMORY=`echo $USED_MEMORY/1024/1024 | bc`
sed -i '${s/$/, '$USED_MEMORY'/}' $OUTPUT_FILE
read -r -a MEM <<< "$(free -m | tr -s " " | grep Mem)"
FREE_MEM=`echo "(${MEM[1]}*85/100-${MEM[2]}-${MEM[4]})" | bc`
i=$((i+STEP))
done
rm a.out

exit 0

```

01.c

```

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

//~ using namespace std;

int main(int argc, char *argv[]) {
    unsigned long n, i, j, k;
    int ** a, * b, * c;
    clock_t t1, t2;
    double dur;
    if (argc > 1) {
        sscanf(argv[1], "%lu", &n);
    } else scanf("%lu", &n);
    t1 = clock();
    a = (int**) calloc(n, sizeof(int*));
    b = (int*) calloc(n, sizeof(int));
    c = (int*) calloc(n, sizeof(int));
    for (i=0; i<n; i++) {
        a[i] = (int*) calloc(n, sizeof(int));
        b[i] = rand()%51-25;
        c[i]=0;
        for (j=0; j<n; j++)
            a[i][j]=rand()%51-25;
    }
    for (i=0; i<n; i++) {
        //~ c[i]=0;
        for (k=0; k<n; k++) {
            c[i]+=a[i][k]*b[k];
        }
    }
    //~ for (i=0; i<n; i++) {
    //~     for (j=0; j<n; j++)
    //~         printf("%li ", a[i][j]);
    //~     printf("\n");
    //~ }
    //~ printf("\n\n");
    //~ for (i=0; i<n; i++) {
    //~     printf("%li ", b[i]);
    //~ }
    //~ printf("\n\n");
    //~ for (i=0; i<n; i++) {
    //~     printf("%li ", c[i]);
    //~ }
}

```

```

    //~ }
    //~ printf("\n\n");
    t2 = clock();
    dur = 1.0*(t2-t1)/CLOCKS_PER_SEC;
    printf("%lu,%lf\n",n,dur);
    for (i=0;i<n;i++)
        free(a[i]);
    free(a);
    return 0;
}

```

02.c

```

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

//~ using namespace std;

int main(int argc,char *argv[]) {
    unsigned long n,i,j,k;
    long long ** a, * b, * c;
    clock_t t1,t2;
    double dur;
    if (argc > 1) {
        sscanf(argv[1],"%lu",&n);
    } else scanf("%lu",&n);
    t1 = clock();
    a = (long long**) calloc(n,sizeof(long long*));
    b = (long long*) calloc(n,sizeof(long long));
    c = (long long*) calloc(n,sizeof(long long));
    for (i=0;i<n;i++) {
        a[i] = (long long*) calloc(n,sizeof(long long));
        b[i] = rand()%51-25;
        c[i]=0;
        for(j=0;j<n;j++)
            a[i][j]=rand()%51-25;
    }
    for (i=0;i<n;i++) {
        //~ c[i]=0;
        for (k=0;k<n;k++) {
            c[i]+=a[i][k]*b[k];
        }
    }
    //~ for (i=0;i<n;i++) {
    //~ for (j=0;j<n;j++)
    //~ printf("%li ",a[i][j]);
    //~ printf("\n");
    //~ }
    //~ printf("\n\n");
    //~ for (i=0;i<n;i++) {
    //~ printf("%li ",b[i]);
    //~ }
    //~ printf("\n\n");
    //~ for (i=0;i<n;i++) {
    //~ printf("%li ",c[i]);
    //~ }
    //~ printf("\n\n");
}

```

```

    t2 = clock();
    dur = 1.0*(t2-t1)/CLOCKS_PER_SEC;
    printf("%lu,%lf\n",n,dur);
    for (i=0;i<n;i++)
        free(a[i]);
    free(a);
    return 0;
}

```

03.c

```

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

//~ using namespace std;

int main(int argc,char *argv[]) {
    unsigned long n,i,j,k;
    long double ** a, * b, * c;
    clock_t t1,t2;
    double dur;
    if (argc > 1) {
        sscanf(argv[1],"%lu",&n);
    } else scanf("%lu",&n);
    t1 = clock();
    a = (long double**) calloc(n,sizeof(long double*));
    b = (long double*) calloc(n,sizeof(long double));
    c = (long double*) calloc(n,sizeof(long double));
    for (i=0;i<n;i++) {
        a[i] = (long double*) calloc(n,sizeof(long double));
        b[i] = rand()%51-25;
        c[i]=0;
        for(j=0;j<n;j++)
            a[i][j]=rand()%51-25;
    }
    for (i=0;i<n;i++) {
        //~ c[i]=0;
        for (k=0;k<n;k++) {
            c[i]+=a[i][k]*b[k];
        }
    }
    //~ for (i=0;i<n;i++) {
    //~ for (j=0;j<n;j++)
    //~ printf("%li ",a[i][j]);
    //~ printf("\n");
    //~ }
    //~ printf("\n\n");
    //~ for (i=0;i<n;i++) {
    //~ printf("%li ",b[i]);
    //~ }
    //~ printf("\n\n");
    //~ for (i=0;i<n;i++) {
    //~ printf("%li ",c[i]);
    //~ }
    //~ printf("\n\n");
    t2 = clock();
    dur = 1.0*(t2-t1)/CLOCKS_PER_SEC;

```

```

        printf("%lu,%lf\n",n,dur);
        for (i=0;i<n;i++)
            free(a[i]);
        free(a);
        return 0;
}

```

04.c

```

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

//~ using namespace std;

int main(int argc,char *argv[]) {
    unsigned long n,i,j,k;
    int ** a, * b, * c;
    clock_t t1,t2;
    double dur;
    if (argc > 1) {
        sscanf(argv[1],"%lu",&n);
    } else scanf("%lu",&n);
    t1 = clock();
    a = (int**) calloc(n,sizeof(int*));
    b = (int*) calloc(n,sizeof(int));
    c = (int*) calloc(n,sizeof(int));
    for (i=0;i<n;i++) {
        a[i] = (int*) calloc(n,sizeof(int));
        b[i] = rand()%51-25;
        c[i]=0;
        for(j=0;j<n;j++)
            a[i][j]=rand()%51-25;
    }
    for (i=0;i<n;i++) {
        //~ c[i]=0;
        for (k=0;k<n;k++) {
            c[k]+=a[k][i]*b[i];
        }
    }
    //~ for (i=0;i<n;i++) {
    //~     for (j=0;j<n;j++)
    //~         printf("%li ",a[i][j]);
    //~     printf("\n");
    //~ }
    //~ printf("\n\n");
    //~ for (i=0;i<n;i++) {
    //~     printf("%li ",b[i]);
    //~ }
    //~ printf("\n\n");
    //~ for (i=0;i<n;i++) {
    //~     printf("%li ",c[i]);
    //~ }
    //~ printf("\n\n");
    t2 = clock();
    dur = 1.0*(t2-t1)/CLOCKS_PER_SEC;
    printf("%lu,%lf\n",n,dur);
    for (i=0;i<n;i++)

```

```

        free(a[i]);
    free(a);
    return 0;
}

```

05.c

```

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

//~ using namespace std;

int main(int argc, char *argv[]) {
    unsigned long n, i, j, k;
    long long ** a, * b, * c;
    clock_t t1, t2;
    double dur;
    if (argc > 1) {
        sscanf(argv[1], "%lu", &n);
    } else scanf("%lu", &n);
    t1 = clock();
    a = (long long**) calloc(n, sizeof(long long*));
    b = (long long*) calloc(n, sizeof(long long));
    c = (long long*) calloc(n, sizeof(long long));
    for (i=0; i<n; i++) {
        a[i] = (long long*) calloc(n, sizeof(long long));
        b[i] = rand()%51-25;
        c[i]=0;
        for(j=0; j<n; j++)
            a[i][j]=rand()%51-25;
    }
    for (i=0; i<n; i++) {
        //~ c[i]=0;
        for (k=0; k<n; k++) {
            c[k]+=a[k][i]*b[i];
        }
    }
    //~ for (i=0; i<n; i++) {
    //~ for (j=0; j<n; j++)
    //~ printf("%li ", a[i][j]);
    //~ printf("\n");
    //~ }
    //~ printf("\n\n");
    //~ for (i=0; i<n; i++) {
    //~ printf("%li ", b[i]);
    //~ }
    //~ printf("\n\n");
    //~ for (i=0; i<n; i++) {
    //~ printf("%li ", c[i]);
    //~ }
    //~ printf("\n\n");
    t2 = clock();
    dur = 1.0*(t2-t1)/CLOCKS_PER_SEC;
    printf("%lu, %lf\n", n, dur);
    for (i=0; i<n; i++)
        free(a[i]);
    free(a);
}

```

```

    return 0;
}

```

06.c

```

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

//~ using namespace std;

int main(int argc, char *argv[]) {
    unsigned long n, i, j, k;
    long double ** a, * b, * c;
    clock_t t1, t2;
    double dur;
    if (argc > 1) {
        sscanf(argv[1], "%lu", &n);
    } else scanf("%lu", &n);
    t1 = clock();
    a = (long double**) calloc(n, sizeof(long double*));
    b = (long double*) calloc(n, sizeof(long double));
    c = (long double*) calloc(n, sizeof(long double));
    for (i=0; i<n; i++) {
        a[i] = (long double*) calloc(n, sizeof(long double));
        b[i] = rand()%51-25;
        c[i]=0;
        for (j=0; j<n; j++)
            a[i][j]=rand()%51-25;
    }
    for (i=0; i<n; i++) {
        //~ c[i]=0;
        for (k=0; k<n; k++) {
            c[k]+=a[k][i]*b[i];
        }
    }
    //~ for (i=0; i<n; i++) {
    //~ for (j=0; j<n; j++)
    //~ printf("%li ", a[i][j]);
    //~ printf("\n");
    //~ }
    //~ printf("\n\n");
    //~ for (i=0; i<n; i++) {
    //~ printf("%li ", b[i]);
    //~ }
    //~ printf("\n\n");
    //~ for (i=0; i<n; i++) {
    //~ printf("%li ", c[i]);
    //~ }
    //~ printf("\n\n");
    t2 = clock();
    dur = 1.0*(t2-t1)/CLOCKS_PER_SEC;
    printf("%lu, %lf\n", n, dur);
    for (i=0; i<n; i++)
        free(a[i]);
    free(a);
    return 0;
}

```


07.c

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

//~ using namespace std;

int main(int argc, char *argv[]) {
    unsigned long n, i, j, k;
    int ** a, * b, * c;
    clock_t t1, t2;
    double dur;
    if (argc > 1) {
        sscanf(argv[1], "%lu", &n);
    } else scanf("%lu", &n);
    t1 = clock();
    a = (int**) calloc(n, sizeof(int*));
    b = (int*) calloc(n, sizeof(int));
    c = (int*) calloc(n, sizeof(int));
    for (i=0; i<n; i++) {
        a[i] = (int*) calloc(n, sizeof(int));
        b[i] = rand()%51-25;
        c[i]=0;
        for(j=0; j<n; j++)
            a[i][j]=rand()%51-25;
    }
    for (i=0; i<n; i++) {
        for (k=0; k<n; k++) {
            c[k]+=b[i]*a[i][k];
        }
    }
    //~ for (i=0; i<n; i++) {
    //~     for (j=0; j<n; j++)
    //~         printf("%li ", a[i][j]);
    //~     printf("\n");
    //~ }
    //~ printf("\n\n");
    //~ for (i=0; i<n; i++) {
    //~     printf("%li ", b[i]);
    //~ }
    //~ printf("\n\n");
    //~ for (i=0; i<n; i++) {
    //~     printf("%li ", c[i]);
    //~ }
    //~ printf("\n\n");
    t2 = clock();
    dur = 1.0*(t2-t1)/CLOCKS_PER_SEC;
    printf("%lu, %lf\n", n, dur);
    for (i=0; i<n; i++)
        free(a[i]);
    free(a);
    return 0;
}
```

08.c

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

```

#include <stdlib.h>
#include <time.h>

//~ using namespace std;

int main(int argc, char *argv[]) {
    unsigned long n,i,j,k;
    long long ** a, * b, * c;
    clock_t t1,t2;
    double dur;
    if (argc > 1) {
        sscanf(argv[1], "%lu", &n);
    } else scanf("%lu", &n);
    t1 = clock();
    a = (long long**) calloc(n, sizeof(long long*));
    b = (long long*) calloc(n, sizeof(long long));
    c = (long long*) calloc(n, sizeof(long long));
    for (i=0; i<n; i++) {
        a[i] = (long long*) calloc(n, sizeof(long long));
        b[i] = rand()%51-25;
        c[i]=0;
        for(j=0; j<n; j++)
            a[i][j]=rand()%51-25;
    }
    for (i=0; i<n; i++) {
        for (k=0; k<n; k++) {
            c[k]+=b[i]*a[i][k];
        }
    }
    //~ for (i=0; i<n; i++) {
    //~     for (j=0; j<n; j++)
    //~         printf("%li ", a[i][j]);
    //~     printf("\n");
    //~ }
    //~ printf("\n\n");
    //~ for (i=0; i<n; i++) {
    //~     printf("%li ", b[i]);
    //~ }
    //~ printf("\n\n");
    //~ for (i=0; i<n; i++) {
    //~     printf("%li ", c[i]);
    //~ }
    //~ printf("\n\n");
    t2 = clock();
    dur = 1.0*(t2-t1)/CLOCKS_PER_SEC;
    printf("%lu, %lf\n", n, dur);
    for (i=0; i<n; i++)
        free(a[i]);
    free(a);
    return 0;
}

```

09.c

```

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

```

```

//~ using namespace std;

int main(int argc, char *argv[]) {
    unsigned long n, i, j, k;
    long double ** a, * b, * c;
    clock_t t1, t2;
    double dur;
    if (argc > 1) {
        sscanf(argv[1], "%lu", &n);
    } else scanf("%lu", &n);
    t1 = clock();
    a = (long double**) calloc(n, sizeof(long double*));
    b = (long double*) calloc(n, sizeof(long double));
    c = (long double*) calloc(n, sizeof(long double));
    for (i=0; i<n; i++) {
        a[i] = (long double*) calloc(n, sizeof(long double));
        b[i] = rand()%51-25;
        c[i]=0;
        for(j=0; j<n; j++)
            a[i][j]=rand()%51-25;
    }
    for (i=0; i<n; i++) {
        for (k=0; k<n; k++) {
            c[k]+=b[i]*a[i][k];
        }
    }
    //~ for (i=0; i<n; i++) {
    //~     for (j=0; j<n; j++)
    //~         printf("%li ", a[i][j]);
    //~     printf("\n");
    //~ }
    //~ printf("\n\n");
    //~ for (i=0; i<n; i++) {
    //~     printf("%li ", b[i]);
    //~ }
    //~ printf("\n\n");
    //~ for (i=0; i<n; i++) {
    //~     printf("%li ", c[i]);
    //~ }
    //~ printf("\n\n");
    t2 = clock();
    dur = 1.0*(t2-t1)/CLOCKS_PER_SEC;
    printf("%lu, %lf\n", n, dur);
    for (i=0; i<n; i++)
        free(a[i]);
    free(a);
    return 0;
}

```