МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет информационных технологий

Кафедра параллельных вычислений

ОТЧЕТ

О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

«Определение времени работы прикладных программ»

Студента 2 курса, 21211 группы

Петрова Сергея Евгеньевича

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:
Антон Юрьевич Кудинов

Новосибирск 2022

СОДЕРЖАНИЕ

ЦЕЛЬ

- *Изучение методики измерения времени работы подпрограммы;*
- *Изучение приемов повышения точности измерения времени работы подпрограммы;*
- *Изучение способов измерения времени работы подпрограммы;*
- *Измерение времени работы подпрограммы в прикладной программе;*

ЗАДАНИЕ

1. *Написать программу на языке C или C++, которая реализует алгоритм вычисления числа Пи метом Монте-Карло. Алгоритм состоит в следующем. Сначала в квадрат с центром в начале координат и со стороной два вписывается круг с единичным радиусом. Затем в этом квадрате случайным образом с равномерным распределением генерируются N точек. Точка может попасть в окружность или нет (условие попадания $x^2 + y^2 \leq 1$). Далее определяется число M точек, попавших в круг. При достаточно большом числе бросков N, по значениям M и N вычисляется число Пи:*

$$\pi \approx \frac{4M}{N}$$

2. *Проверить правильность работы программы на нескольких тестовых наборах входных данных;*
3. *Выбрать значение параметра N таким, чтобы время работы программы было порядка 15 секунд;*
4. *По приведенной методике определить время работы подпрограммы тестовой программы с относительной погрешностью не более 1%.*
5. *Составить отчет по лабораторной работе. Отчет должен содержать следующее:*
    1) *Титульный лист.*
    2) *Цель лабораторной работы.*
    3) *Вариант задания.*
    4) *Описание методики для определения времени работы программы.*
    5) *Результат измерения времени работы программы.*
    6) *Полный компилируемый листинг реализованной программы и команду для ее компиляции.*
    7) *Вывод по результатам лабораторной работы.*

ОПИСАНИЕ РАБОТЫ

Пошаговое описание выполненной работы

1. *Выбрал значение параметра N таким, чтобы время работы программы было порядка 15 секунд. Время программы измерял с помощью утилиты* ***time****.*

Сборка и запуск программы с утилитой time

```
evmpu@comrade:~/21211/s.petrov1/lab1$ cmake -B bin -S src -DCMAKE_BUILD_TYPE=Release
-- The CXX compiler identification is GNU 9.4.0
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/evmpu/21211/s.petrov1/lab1/bin
evmpu@comrade:~/21211/s.petrov1/lab1$ cmake --build bin
Scanning dependencies of target random
[ 12%] Building CXX object monte_carlo/point/random/CMakeFiles/random.dir/random.cpp.o
[ 25%] Linking CXX static library librandom.a
[ 25%] Built target random
Scanning dependencies of target point
[ 37%] Building CXX object monte_carlo/point/CMakeFiles/point.dir/point.cpp.o
[ 50%] Linking CXX static library libpoint.a
[ 50%] Built target point
Scanning dependencies of target monte_carlo
[ 62%] Building CXX object monte_carlo/CMakeFiles/monte_carlo.dir/monte_carlo.cpp.o
[ 75%] Linking CXX static library libmonte_carlo.a
[ 75%] Built target monte_carlo
Scanning dependencies of target lab1
[ 87%] Building CXX object CMakeFiles/lab1.dir/main.cpp.o
[100%] Linking CXX executable lab1
[100%] Built target lab1
evmpu@comrade:~/21211/s.petrov1/lab1$ time bin/lab1 500000000
PI: 3.14154

real    0m17,470s
user    0m17,459s
sys     0m0,008s
```

2. *С помощью команды* ***top*** *оценил степень загрузки процессора другими процессами. Степень загрузки оказался невысокой, поэтому использовать таймер времени процесса нет необходимости.*

Результат команды top

```
Tasks: 434 total,   1 running, 431 sleeping,   2 stopped,   0 zombie
%Cpu(s):  2,4 us,  0,0 sy,  0,0 ni, 97,2 id,  0,3 wa,  0,0 hi,  0,0 si,  0,0 st
МиБ Mem :  24064,6 total,  18504,5 free,   1034,3 used,   4525,7 buff/cache
МиБ Swap:   7813,0 total,   7813,0 free,      0,0 used.  22631,3 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
2688962 evmpu     20   0   11164   5640   3684 S   0,3   0,0   0:00.28 bash
2698433 evmpu     20   0   13992   6124   4644 S   0,3   0,0   0:00.05 sshd
2707744 root      20   0       0      0      0 I   0,3   0,0   0:00.34 kworker/u66:2-events_freezable_power_
2720796 evmpu     20   0   12220   4268   3188 R   0,3   0,0   0:00.06 top
      1 root      20   0  169924  13264   8340 S   0,0   0,1   8:07.26 systemd
      2 root      20   0       0      0      0 S   0,0   0,0   0:00.24 kthreadd
      3 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_gp
```

3. *Использование счетчика тактов процессора не имеет смысл, потому что интервал времени больше кванта времени, выделяемого процессу операционной системой.*
4. *Оценил относительную точность таймера системного времени. Зная, что абсолютная точность* ***clock_gettime*** *равна 1 наносекунд и общее время работы программы - 15 секунд, получил относительную точность равную* $\approx 6.7 \times 10^{-9}$ *%. Таким образом, использование таймера системного времени обеспечивает нужную точность вычисления.*
5. *Перед запуском программы с измерением системного времени выполнил команду* ***sync****, чтобы сгрузить накопленные в буфере отложенной записи данные на диск.*

Строки компиляции и запуска программы

Команды для компиляции и запуска программы с измерением системного времени

```
evmpu@comrade:~/21211/s.petrov1/lab1$ cmake -B bin -S src
-DCMAKE_BUILD_TYPE=Release -DSYS_TIME=true

evmpu@comrade:~/21211/s.petrov1/lab1$ cmake --build bin

evmpu@comrade:~/21211/s.petrov1/lab1$ bin/lab1
```

Команды для компиляции и запуска программы с измерением времени процесса

```
evmpu@comrade:~/21211/s.petrov1/lab1$ cmake -B bin -S src
-DCMAKE_BUILD_TYPE=Release -DPROC_TIME=true

evmpu@comrade:~/21211/s.petrov1/lab1$ cmake --build bin

evmpu@comrade:~/21211/s.petrov1/lab1$ bin/lab1
```

Команды для компиляции и запуска программы с измерением времени счётчиком тактов процессора

```
evmpu@comrade:~/21211/s.petrov1/lab1$ cmake -B bin -S src
-DCMAKE_BUILD_TYPE=Release -DCPU_TIME_STAMP_COUNTER=true

evmpu@comrade:~/21211/s.petrov1/lab1$ cmake --build bin

evmpu@comrade:~/21211/s.petrov1/lab1$ bin/lab1
```

Результат измерения времени работы программы

Результат измерения системного времени

```
evmpu@comrade:~/21211/s.petrov1/lab1$ cmake -Bbin -Ssrc -DCMAKE_BUILD_TYPE=Release -DSYS_TIME=true
-- The CXX compiler identification is GNU 9.4.0
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Macro SYS_TIME installed
-- Library rt installed
-- Configuring done
-- Generating done
-- Build files have been written to: /home/evmpu/21211/s.petrov1/lab1/bin
evmpu@comrade:~/21211/s.petrov1/lab1$ cmake --build bin
Scanning dependencies of target random
[ 12%] Building CXX object monte_carlo/point/random/CMakeFiles/random.dir/random.cpp.o
[ 25%] Linking CXX static library librandom.a
[ 25%] Built target random
Scanning dependencies of target point
[ 37%] Building CXX object monte_carlo/point/CMakeFiles/point.dir/point.cpp.o
[ 50%] Linking CXX static library libpoint.a
[ 50%] Built target point
Scanning dependencies of target monte_carlo
[ 62%] Building CXX object monte_carlo/CMakeFiles/monte_carlo.dir/monte_carlo.cpp.o
[ 75%] Linking CXX static library libmonte_carlo.a
[ 75%] Built target monte_carlo
Scanning dependencies of target lab1
[ 87%] Building CXX object CMakeFiles/lab1.dir/main.cpp.o
[100%] Linking CXX executable lab1
[100%] Built target lab1
evmpu@comrade:~/21211/s.petrov1/lab1$ sync
evmpu@comrade:~/21211/s.petrov1/lab1$ bin/lab1 500000000
System time: 17.4342 sec.
PI: 3.14165
```

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были изучены:

- *Методики измерения времени работы подпрограммы;*
- *Приемы повышения точности измерения времени работы подпрограммы;*
- *Способы измерения времени работы подпрограммы.*

По результатам проведённых исследований можно сделать следующие выводы:

- *Выбор того или иного метода измерения времени программы зависит от требуемой абсолютной и/или относительной точности, длины временного интервала, степени загрузки процессора другими процессами;*
- *Важно знать способы увеличения точности и обращать внимание на влияние следующих факторов: стадии инициализации и завершения программы и ситуации, когда во время работы программы операционная система решит сгрузить накопленные в буфере отложенной записи данные на диск.*

ПРИЛОЖЕНИЕ 1 (ПОЛНЫЙ ЛИСТИНГ ПРОГРАММЫ)

src / main.cpp

```
#include <iostream>
#include "monte_carlo.h"

#ifdef SYS_TIME
    #include <ctime>
#endif // SYS_TIME

#ifdef PROC_TIME
    #include <sys/times.h>
    #include <unistd.h>
#endif // PROC_TIME

#ifdef CPU_TIME_STAMP_COUNTER
    #define CPU_HZ 2100000000ULL
#endif // CPU_TIME_STAMP_COUNTER

using namespace std;

int main(int argc, char **argv)
{
    if (argc == 1)
    {
        cerr << "No point count\n";
        return EXIT_FAILURE;
    }

    long long count = atoll(argv[1]);

    if (count < 0)
    {
        cerr << "Wrong point count\n";
        return EXIT_FAILURE;
    }

#ifdef SYS_TIME
    struct timespec sysStart, sysEnd;
    clock_gettime(CLOCK_MONOTONIC_RAW, &sysStart);
#endif // SYS_TIME

#ifdef PROC_TIME
    struct tms procStart, procEnd;
    long clocks_per_sec = sysconf(_SC_CLK_TCK);
    long clocks;
    times(&procStart);
#endif // PROC_TIME

#ifdef CPU_TIME_STAMP_COUNTER
    union
    {
        unsigned long long t64;
        struct
        {
            unsigned long th, tl;
        } t32;
    } tactStart, tactEnd;
    asm("rdtsc\n":"=a"(tactStart.t32.th),"=d"(tactStart.t32.tl));
#endif // CPU_TIME_STAMP_COUNTER
```

```
    double pi = MonteCarloAlgorithm(count);

#ifdef SYS_TIME
    clock_gettime(CLOCK_MONOTONIC_RAW, &sysEnd);
    double sysTime = sysEnd.tv_sec - sysStart.tv_sec + 1e-9 *
(sysEnd.tv_nsec - sysStart.tv_nsec);
    cout << "System time: " << sysTime << " sec.\n";
#endif // SYS_TIME

#ifdef PROC_TIME
    times(&procEnd);
    double procTime = (double)(procEnd.tms_utime - procStart.tms_utime) /
clocks_per_sec;
    cout << "Process time: " << procTime << "sec.\n";
#endif // PROC_TIME

#ifdef CPU_TIME_STAMP_COUNTER
    asm("rdtsc\n":"=a"(tactEnd.t32.th),"=d"(tactEnd.t32.tl));
    double tactTime = (double)(tactEnd.t64 - tactStart.t64) / CPU_HZ;
    cout << "CPU time stamp counter: " << tactTime << " sec.\n";
#endif // CPU_TIME_STAMP_COUNTER

    cout << "PI: " << pi << "\n";

    return EXIT_SUCCESS;
}
```

src / CMakeLists.txt

```
cmake_minimum_required(VERSION 3.5.1)
project(lab1 CXX)

add_executable(lab1 main.cpp)

add_subdirectory(monte_carlo)
target_include_directories(lab1 PUBLIC monte_carlo)
target_link_libraries(lab1 PUBLIC monte_carlo)

if(SYS_TIME)
    message(STATUS "Macro SYS_TIME installed")
    target_compile_definitions (lab1 PUBLIC SYS_TIME)

    find_library(LIBRT rt)
    if(LIBRT)
        message(STATUS "Library rt installed")
        target_link_libraries(lab1 PUBLIC ${LIBRT})
    else()
        message(STATUS "Library rt skipped")
    endif()
endif()
if(PROC_TIME)
    message(STATUS "Macro PROC_TIME installed")
    target_compile_definitions (lab1 PUBLIC PROC_TIME)
endif()
if(CPU_TIME_STAMP_COUNTER)
    message(STATUS "Macro CPU_TIME_STAMP_COUNTER installed")
    target_compile_definitions (lab1 PUBLIC CPU_TIME_STAMP_COUNTER)
endif()
```

src / monte_carlo / monte_carlo.cpp

```cpp
#include "monte_carlo.h"

double MonteCarloAlgorithm(long long count)
{
    InitRand();
    double insideCount = 0.0;
    for (long long i = 0; i < count; ++i)
    {
        Point a;
        if (a.InsideCircle())
        {
            insideCount += 4.0;
        }
    }

    return insideCount / count;
}
```

src / monte_carlo / monte_carlo.h

```cpp
#ifndef MONTE_CARLO_H
#define MONTE_CARLO_H

#include "point.h"
#include "random.h"

double MonteCarloAlgorithm(long long);

#endif // MONTE_CARLO_H
```

src / monte_carlo / CMakeLists.txt

```
add_library(monte_carlo STATIC monte_carlo.cpp)
target_include_directories(monte_carlo PUBLIC point)
add_subdirectory(point)
target_link_libraries(monte_carlo PUBLIC point)
```

src / monte_carlo / point / point.cpp

```cpp
#include "point.h"

Point::Point()
{
    x = GenerateRand();
    y = GenerateRand();
}

bool Point::InsideCircle()
{
    return (x * x) + (y * y) <= 1.0;
}
```

src / monte_carlo / point / point.h

```
#ifndef POINT_H
#define POINT_H

#include "random.h"

class Point
{
public:
    Point();
    bool InsideCircle();
private:
    double x, y;
};

#endif // POINT_H
```

src / monte_carlo / point / CMakeLists.txt

```
add_library(point STATIC point.cpp)
target_include_directories(point PUBLIC random)
add_subdirectory(random)
target_link_libraries(point PUBLIC random)
```

src / monte_carlo / point / random / random.cpp

```
#include "random.h"

void InitRand()
{
    srand(time(NULL));
}

double GenerateRand()
{
    return (double)rand() / RAND_MAX;
}
```

src / monte_carlo / point / random / random.h

```
#ifndef RANDOM_H
#define RANDOM_H

#include <cstdlib>
#include <ctime>

void InitRand();
double GenerateRand();

#endif // RANDOM_H
```

src / monte_carlo / point / random / CMakeLists.txt

```
add_library(random STATIC random.cpp)
```