**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра теории вероятностей и кибербезопасности**

**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**

*дисциплина: Параллельное программирование*

Студент: Добычин Дмитрий Романович

Студ. билет № 1032217059

Группа: НПИбд-01-21

**МОСКВА**

2024 г.

# Общая информация:

# Язык программирования: Fortran

**Задание №1:**

* Используйте подпрограмму random\_number и напишите функцию, которая возвращает случайное целое число 𝑖, такое что 1 ⩽ 𝑖 ⩽ 𝑁.
* Проверьте на примере 𝑛 = 6, что вероятность выпадения чисел 1, 2, 3, 4, 5, 6 одинакова, то есть распределение равномерное. Это соответствует задаче по подбрасыванию шестигранного игрального кубика.
* Проверку осуществите, проведя минимум 𝑁 = 106 испытаний методом Монте-Карло.
* Программу ускорьте с помощью параллельных потоков OpenMP.

**Реализация на Fortran задания №1:**

**program** monte\_carlo\_dice

**use** omp\_lib

**implicit** **none**

**integer,** **parameter** **::** N **=** 6

**integer,** **parameter** **::** num\_trials **=** 1e6

**integer** **::** i**,** rand\_val**,** chunk**,** num\_threads

**integer** **::** counts**(**N**)**

**real** **::** rnum

**integer** **::** seed\_size

**integer,** **dimension(:),** **allocatable** **::** seed

! Узнаем размер массива для seed

**call** **random\_seed(size=**seed\_size**)**

**allocate(**seed**(**seed\_size**))**

! Инициализация генератора случайных чисел

seed **=** **[(**i**,** i **=** 1**,** seed\_size**)]**

**call** **random\_seed(**put**=**seed**)**

! Инициализируем массив для подсчета выпадений на 0

counts **=** 0

! Параллельная секция с использованием OpenMP

!$omp parallel private(i, rand\_val, rnum) shared(counts)

!$omp do

**do** i **=** 1**,** num\_trials

! Генерация случайного числа от 1 до N

**call** **random\_number(**rnum**)**

rand\_val **=** **int(**rnum **\*** N**)** **+** 1

! Критическая секция для защиты общего ресурса counts

!$omp atomic

counts**(**rand\_val**)** **=** counts**(**rand\_val**)** **+** 1

**end** **do**

!$omp end do

!$omp end parallel

! Вывод результатов

**print** **\*,** **"Результаты подбрасывания кубика:"**

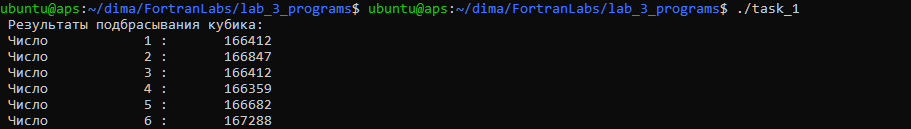
**do** i **=** 1**,** N

**print** **\*,** **"Число ",** i**,** **": ",** counts**(**i**)**

**end** **do**

**deallocate(**seed**)**

**end** **program** monte\_carlo\_dice

****

**Рис. 1.** Результат работы программы для задания №1.

**Задание №2:**

Смоделируйте задачу про парадокс Монти Холла методом Монте-Карло для двух стратегий игрока:

1. игрок никогда не меняет свое первое решение и открывает ту дверь, которую выбрал первой;
2. игрок всегда меняет свое решение и открывает вторую дверь, предложенную Монти Холлом.

Программа должна работать в параллельном режиме. Вычислите частоту побед для обеих стратегий. Теоретическое решение дает вероятность 2/3 выиграть при смене решения и при 1/3 если решение не менять.

**Реализация на Fortran задания №2:**

**program** monty\_hall

**use** omp\_lib

**implicit** **none**

**integer,** **parameter** **::** num\_trials **=** 1e6

**integer** **::** win\_no\_change **=** 0**,** win\_change **=** 0

**integer** **::** i**,** prize\_door**,** player\_choice**,** revealed\_door**,** change\_choice

**real** **::** rnum

**integer** **::** seed\_size

**integer,** **dimension(:),** **allocatable** **::** seed

! Узнаем размер массива для seed

**call** **random\_seed(size=**seed\_size**)**

**allocate(**seed**(**seed\_size**))**

! Инициализация генератора случайных чисел

seed **=** **[(**i**,** i **=** 1**,** seed\_size**)]**

**call** **random\_seed(**put**=**seed**)**

! Параллельная секция OpenMP

!$omp parallel private(i, prize\_door, player\_choice, revealed\_door, change\_choice, rnum) shared(win\_no\_change, win\_change)

!$omp do reduction(+:win\_no\_change, win\_change)

**do** i **=** 1**,** num\_trials

! 1. Случайный выбор двери с машиной

**call** **random\_number(**rnum**)**

prize\_door **=** **int(**rnum **\*** 3**)** **+** 1 ! Машина за дверью 1, 2 или 3

! 2. Игрок делает случайный выбор двери

**call** **random\_number(**rnum**)**

player\_choice **=** **int(**rnum **\*** 3**)** **+** 1

! 3. Ведущий открывает одну из оставшихся дверей, за которой козёл

**do**

**call** **random\_number(**rnum**)**

revealed\_door **=** **int(**rnum **\*** 3**)** **+** 1

**if** **(**revealed\_door **/=** prize\_door .and. revealed\_door **/=** player\_choice**)** **exit**

**end** **do**

! 4. Стратегия 1: игрок не меняет свой выбор

**if** **(**player\_choice **==** prize\_door**)** **then**

win\_no\_change **=** win\_no\_change **+** 1

**end** **if**

! 5. Стратегия 2: игрок меняет свой выбор

**do**

**call** **random\_number(**rnum**)**

change\_choice **=** **int(**rnum **\*** 3**)** **+** 1

**if** **(**change\_choice **/=** player\_choice .and. change\_choice **/=** revealed\_door**)** **exit**

**end** **do**

**if** **(**change\_choice **==** prize\_door**)** **then**

win\_change **=** win\_change **+** 1

**end** **if**

**end** **do**

!$omp end do

!$omp end parallel

! Вывод результатов

**print** **\*,** **"Количество испытаний: ",** num\_trials

**print** **\*,** **"Побед при стратегии без смены выбора: ",** win\_no\_change

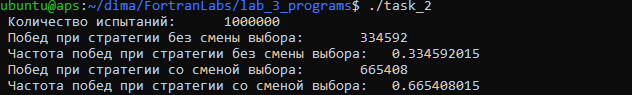
**print** **\*,** **"Частота побед при стратегии без смены выбора: ",** **real(**win\_no\_change**)** **/** num\_trials

**print** **\*,** **"Побед при стратегии со сменой выбора: ",** win\_change

**print** **\*,** **"Частота побед при стратегии со сменой выбора: ",** **real(**win\_change**)** **/** num\_trials

**deallocate(**seed**)**

**end** **program** monty\_hall

****

**Рис. 2.** Результат работы программы для задания №2.

**Задание №3:**

Смоделируйте задачу про парадокс дней рожденья методом Монте-Карло. Вычислите частоту совпадений дней рождения для 106 испытаний. Постройте график изменения этой частоты в зависимости от количества людей в группы. Начните с двух человек (𝑛 = 2) и до 100 человек (𝑛 = 100). Программа должна работать в параллельном режиме.

**Реализация на Fortran задания №3:**

**program** birthday\_paradox

**use** omp\_lib ! Подключаем библиотеку OpenMP для параллельных вычислений

**implicit** **none**

**integer,** **parameter** **::** num\_trials **=** 1000000 ! Количество испытаний (экспериментов)

**integer,** **parameter** **::** max\_people **=** 100 ! Максимальное количество людей в группе

**integer,** **parameter** **::** days\_in\_year **=** 365 ! Количество дней в году

**integer** **::** n**,** i**,** j**,** k**,** trials\_with\_match

**logical** **::** has\_match

**integer** **::** birthdays**(**max\_people**),** count\_matches**(**max\_people**)**

**real** **::** frequency**(**max\_people**)**

**real** **::** rnum

**character(len=**20**)** **::** filename

**integer** **::** unit\_num

! Инициализация начальных значений

count\_matches **=** 0 ! Количество совпадений для каждого значения n

! Параллельная секция для проведения экспериментов

!$omp parallel private(i, j, k, birthdays, has\_match, rnum) shared(count\_matches)

**do** n **=** 2**,** max\_people ! Количество людей в группе от 2 до 100

!$omp do reduction(+:count\_matches)

**do** i **=** 1**,** num\_trials ! Проводим num\_trials испытаний для каждой группы из n человек

birthdays **=** 0 ! Обнуляем массив с днями рождения для нового эксперимента

has\_match **=** .false. ! Флаг совпадения дней рождения

! Генерация случайных дней рождения для группы из n человек

**do** j **=** 1**,** n

**call** **random\_number(**rnum**)**

birthdays**(**j**)** **=** **int(**rnum **\*** days\_in\_year**)** **+** 1 ! Случайный день от 1 до 365

**end** **do**

! Проверка на совпадения

**do** j **=** 1**,** n **-** 1

**do** k **=** j **+** 1**,** n

**if** **(**birthdays**(**j**)** **==** birthdays**(**k**))** **then**

has\_match **=** .true. ! Нашли совпадение

**exit**

**end** **if**

**end** **do**

**if** **(**has\_match**)** **exit**

**end** **do**

**if** **(**has\_match**)** count\_matches**(**n**)** **=** count\_matches**(**n**)** **+** 1 ! Фиксируем совпадение для данной группы

**end** **do**

!$omp end do

**end** **do**

!$omp end parallel

! Открываем файл для записи данных

filename = 'birthday\_results.csv'

unit\_num = 10 ! Номер файла

open(unit=unit\_num, file=filename, status='replace')

! Записываем заголовки в CSV файл

write(unit\_num, \*) 'n,frequency'

! Вычисление частоты совпадений для каждого количества людей и запись в файл

do n = 2, max\_people

frequency(n) = real(count\_matches(n)) / num\_trials

print \*, "Количество людей: ", n, " Частота совпадений: ", frequency(n)

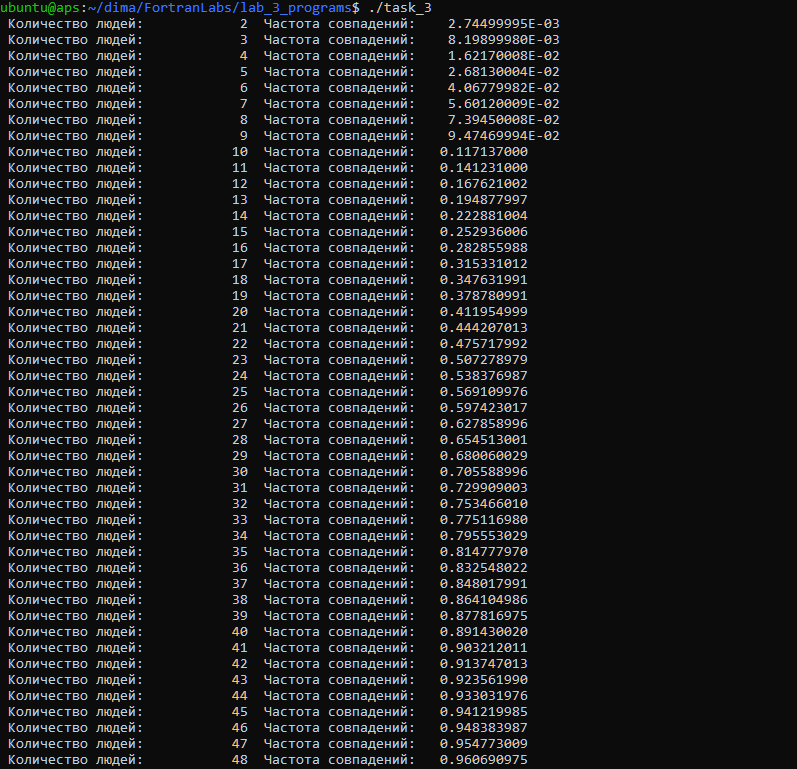
write(unit\_num, '(I3, ",", F7.5)') n, frequency(n) ! Записываем данные в CSV файл

end do

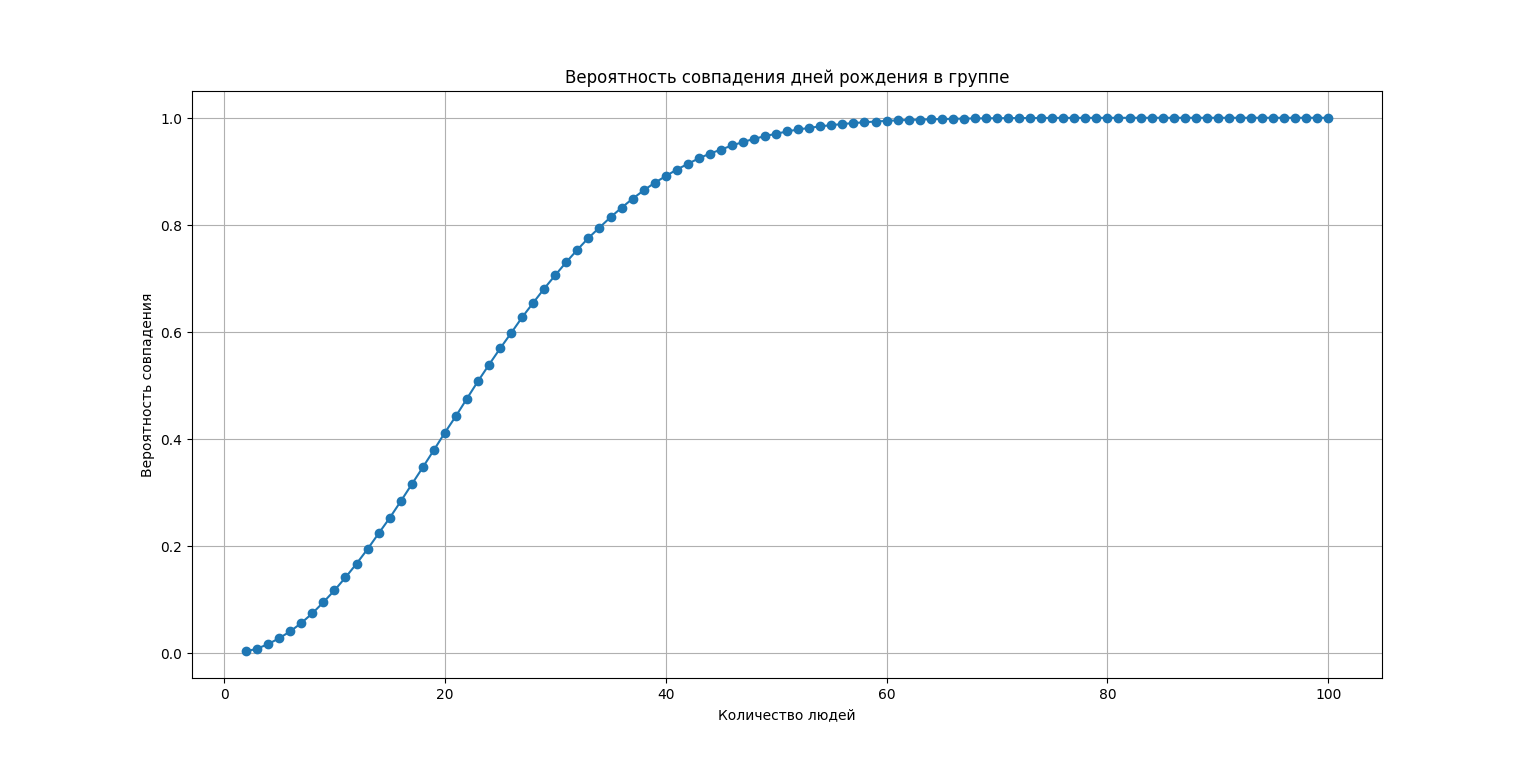
! Закрываем файл

close(unit\_num)

end program birthday\_paradox

****

**Рис. 3.1.** Результат работы программы для задания №3.

****

**Рис. 3.2.** Построение графика в Python.

import matplotlib.pyplot as plt  
import csv  
  
# Чтение данных из CSV файла  
n\_people = []  
frequency = []  
  
with open('birthday\_results.csv', mode='r') as file:  
 reader = csv.reader(file)  
 next(reader) # Пропустить заголовок  
 for row in reader:  
 n\_people.append(int(row[0])) # Преобразуем n к целому  
 frequency.append(float(row[1])) # Преобразуем вероятность к числу с плавающей точкой  
  
# Построение графика  
plt.plot(n\_people, frequency, marker='o')  
plt.title('Вероятность совпадения дней рождения в группе')  
plt.xlabel('Количество людей')  
plt.ylabel('Вероятность совпадения')  
plt.grid(True)  
plt.show()

**Задание №4:**

Смоделируйте задачу о разорении игрока методом Монте-Карло. Вычислите частоту побед и проигрышей. Вычислите также среднее число раундов в каждой игре.

**Реализация на Fortran задания №4:**

**program** gambler\_ruin

**use** omp\_lib ! Подключаем библиотеку OpenMP для параллельных вычислений

**implicit** **none**

**integer,** **parameter** **::** N **=** 100 ! Общий капитал (N рублей)

**integer,** **parameter** **::** num\_trials **=** 1000000 ! Количество игр (испытаний)

**integer** **::** n\_initial**,** current\_capital**,** i**,** round\_count**,** wins**,** losses

**real** **::** p**,** q**,** rnum ! Вероятности и случайное число

**real** **::** avg\_rounds**,** total\_rounds

! Инициализация переменных

n\_initial **=** 90 ! Начальный капитал игрока №1

p **=** 0.5 ! Вероятность выигрыша игрока №1

q **=** 1.0 **-** p ! Вероятность проигрыша игрока №1

wins **=** 0 ! Количество побед игрока №1

losses **=** 0 ! Количество разорений

total\_rounds **=** 0 ! Общее количество раундов

! Параллельное выполнение Монте-Карло

!$omp parallel private(current\_capital, round\_count, i, rnum) shared(wins, losses, total\_rounds)

!$omp do reduction(+:wins, losses, total\_rounds)

**do** i **=** 1**,** num\_trials

current\_capital **=** n\_initial ! Изначальный капитал

round\_count **=** 0

! Игровой процесс (случайное блуждание)

**do** **while** **(**current\_capital **>** 0 .and. current\_capital **<** N**)**

**call** **random\_number(**rnum**)** ! Генерация случайного числа от 0 до 1

**if** **(**rnum **<** p**)** **then**

current\_capital **=** current\_capital **+** 1 ! Игрок выиграл раунд

**else**

current\_capital **=** current\_capital **-** 1 ! Игрок проиграл раунд

**end** **if**

round\_count **=** round\_count **+** 1 ! Увеличиваем количество раундов

**end** **do**

! Проверка исхода игры

**if** **(**current\_capital **==** N**)** **then**

wins **=** wins **+** 1 ! Игрок 1 выиграл

**else**

losses **=** losses **+** 1 ! Игрок 1 разорился

**end** **if**

total\_rounds **=** total\_rounds **+** round\_count ! Суммируем количество раундов

**end** **do**

!$omp end do

!$omp end parallel

! Вычисление среднего числа раундов

avg\_rounds = real(total\_rounds) / num\_trials

! Вывод результатов

print \*, "Количество игр: ", num\_trials

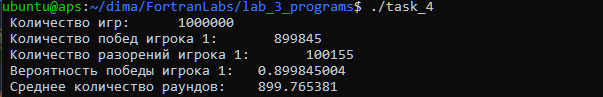
print \*, "Количество побед игрока 1: ", wins

print \*, "Количество разорений игрока 1: ", losses

print \*, "Вероятность победы игрока 1: ", real(wins) / num\_trials

print \*, "Среднее количество раундов: ", avg\_rounds

end program gambler\_ruin

****

**Рис. 4.** Результат работы программы для задания №4.

**Задание №5:**

Смоделируйте игру в жетоны тремя игроками методом Монте-Карло. Вычислите частоту побед и проигрышей. Вычислите среднюю продолжительность раундов в каждой игре и сравните ее с теоретическим значением.

**Реализация на Fortran задания №5:**

**program** token\_game

**use** omp\_lib

**implicit** **none**

**integer,** **parameter** **::** num\_trials **=** 1000000

**integer** **::** x**,** y**,** z

**integer** **::** initial\_x**,** initial\_y**,** initial\_z

**integer** **::** round\_count**,** trial

**integer** **::** winner**,** total\_rounds

**integer** **::** wins\_player1**,** wins\_player2**,** wins\_player3

**real** **::** rnum

**real** **::** theoretical\_duration

! Инициализация начальных жетонов

initial\_x **=** 5

initial\_y **=** 5

initial\_z **=** 5

! Инициализация счетчиков

wins\_player1 **=** 0

wins\_player2 **=** 0

wins\_player3 **=** 0

total\_rounds **=** 0

! Параллельный блок с OpenMP

!$omp parallel private(x, y, z, round\_count, trial, winner, rnum) shared(wins\_player1, wins\_player2, wins\_player3, total\_rounds)

!$omp do reduction(+:wins\_player1, wins\_player2, wins\_player3, total\_rounds)

**do** trial **=** 1**,** num\_trials

! Инициализация жетонов перед каждой игрой

x **=** initial\_x

y **=** initial\_y

z **=** initial\_z

round\_count **=** 0

! Игра продолжается, пока никто не потерял все жетоны

**do** **while** **(**x **>** 0 .and. y **>** 0 .and. z **>** 0**)**

**call** **random\_number(**rnum**)**

winner **=** **int(**3 **\*** rnum**)** **+** 1 ! Бросок кубика (выбираем случайное число от 1 до 3)

! Каждый игрок ставит один жетон

x **=** x **-** 1

y **=** y **-** 1

z **=** z **-** 1

! Победитель забирает жетоны

**if** **(**winner **==** 1**)** **then**

x **=** x **+** 3

**elseif** **(**winner **==** 2**)** **then**

y = y + 3

elseif (winner == 3) then

z = z + 3

end if

round\_count = round\_count + 1

end do

! Определяем победителя

if (x > 0) then

wins\_player1 = wins\_player1 + 1

elseif (y > 0) then

wins\_player2 = wins\_player2 + 1

elseif (z > 0) then

wins\_player3 = wins\_player3 + 1

end if

! Суммируем количество раундов

total\_rounds = total\_rounds + round\_count

end do

!$omp end do

!$omp end parallel

! Теоретическое значение средней продолжительности игры

theoretical\_duration = real(initial\_x \* initial\_y \* initial\_z) / real(initial\_x + initial\_y + initial\_z - 2)

! Вывод результатов

print \*, "Частота побед игрока 1: ", real(wins\_player1) / real(num\_trials)

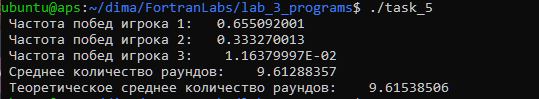
print \*, "Частота побед игрока 2: ", real(wins\_player2) / real(num\_trials)

print \*, "Частота побед игрока 3: ", real(wins\_player3) / real(num\_trials)

print \*, "Среднее количество раундов: ", real(total\_rounds) / real(num\_trials)

print \*, "Теоретическое среднее количество раундов: ", theoretical\_duration

end program token\_game

****

**Рис. 5.** Результат работы программы для задания №5.