

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ  
В.Н.КАРАЗІНА  
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ ТА ПРОГРАМНОГО  
ЗАБЕЗПЕЧЕННЯ

ЛАБОРАТОРНА РОБОТА № 6  
З ДИСЦИПЛІНИ «ОПЕРАЦІЙНІ СИСТЕМИ»

ТЕМА «ПОТОКИ. КЕРУВАННЯ»

Виконав студент 3курсу, групи  
КС31

спеціальності

122 – Комп'ютерні науки

Касьяненко Микита Михайлович

Прийняв:

доцен кафедри шт. ін.

і прогр. забезп. к.н.т.

О.Є. Споров \_\_\_\_

Харків 2022

## Завдання 1

Напишіть програму, яка за допомогою командного рядка отримує параметр — кількість потоків, що створюються. З основної функції викликається інша функція, куди передається робочий масив, у якій створюється задану кількість розрахункових потоків — нащадків і очікується завершення їх роботи. (В спрощеному варіанті програми можна все це зробити в функції `main`.) Кожен розрахунковий потік отримує як аргумент номер цього потоку виконання (відлік починається з 0), засинає на цю кількість секунд і після пробудження генерує псевдовипадкове число в діапазоні  $[1, 10]$ . Потім, отримане псевдовипадкове число заноситься до глобального масиву цілих чисел у комірку, що відповідає номеру потоку, а потік засинає на це згенероване число — кількість секунд. Паралельно з розрахунковими потоками запускається від'єднаний потік, який постійно з періодом 1 секунда виводить на екран повідомлення, що відображає всі ті значення, що на той час зберігаються в масиві. Коли масив стане повністю заповненим, від'єднаний потік самостійно завершує свою роботу. Передбачити інформаційні повідомлення, що пояснюють роботу програми.

## Відповідь

```

mykyta@senku:~/Projects/university-works/operating-systems/laboratory-6/task-1
~/Pr/university-works/o/laboratory-6/task-1 P main +3 !8 ?51 gcc main.c -o main
~/Pr/university-works/o/laboratory-6/task-1 P main +3 !8 ?51 ./main 5

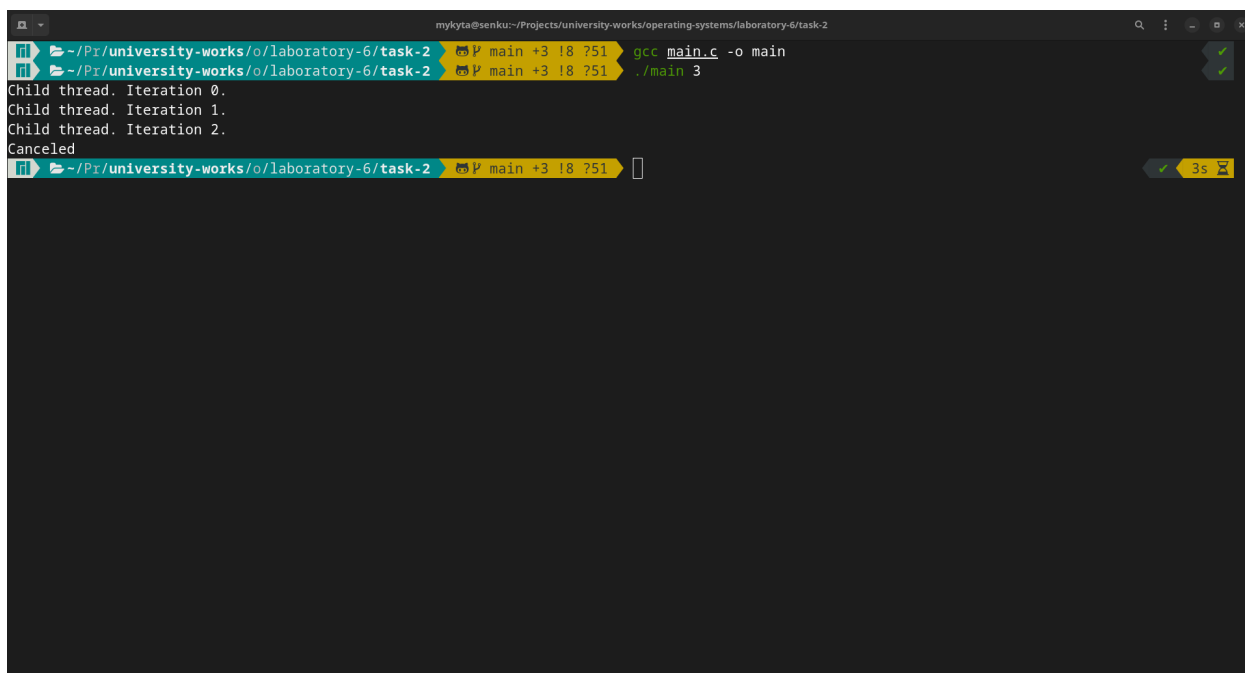
0 Thread sleep 0 second.
Thread sleep 1 second.
0 0 0 0
Thread sleep 4 second.
Thread sleep 3 second.
Thread sleep 2 second.
Discon. thead sleep 1 sec.
Count thread : 0 sleep 5 sec.
Count thread : 1 sleep 5 sec.
5 5 0 0 0
Discon. thead sleep 1 sec.
Count thread : 2 sleep 6 sec.
5 5 6 0 0
Discon. thead sleep 1 sec.
Count thread : 3 sleep 10 sec.
5 5 6 10 0
Discon. thead sleep 1 sec.
Count thread : 4 sleep 4 sec.
Thread 0 finished sucessfully.
Thread 1 finished sucessfully.
Thread 2 finished sucessfully.
Thread 3 finished sucessfully.
Thread 4 finished sucessfully.

```

## Завдання 2

Напишіть програму, яка за допомогою командного рядка отримує параметр – час затримки в секундах. У основній програмі створюється потік – нащадок і очікується завершення роботи. При створенні потік-нащадок встановлюється в режим потоку, що скасовується асинхронно. Потік виконання виводить рядок з текстом (який включає номер ітерації) нескінченну кількість разів і «засинає» на одну секунду після кожного виведення. Після створення потоку основний потік засипає на задану кількість секунд (час затримки) і після пробудження намагається скасувати потік, що працює. Потім аналізується статус завершення потоку і виводиться повідомлення про те, як завершився потік - в результаті скасування або звичайним чином.

## Відповідь

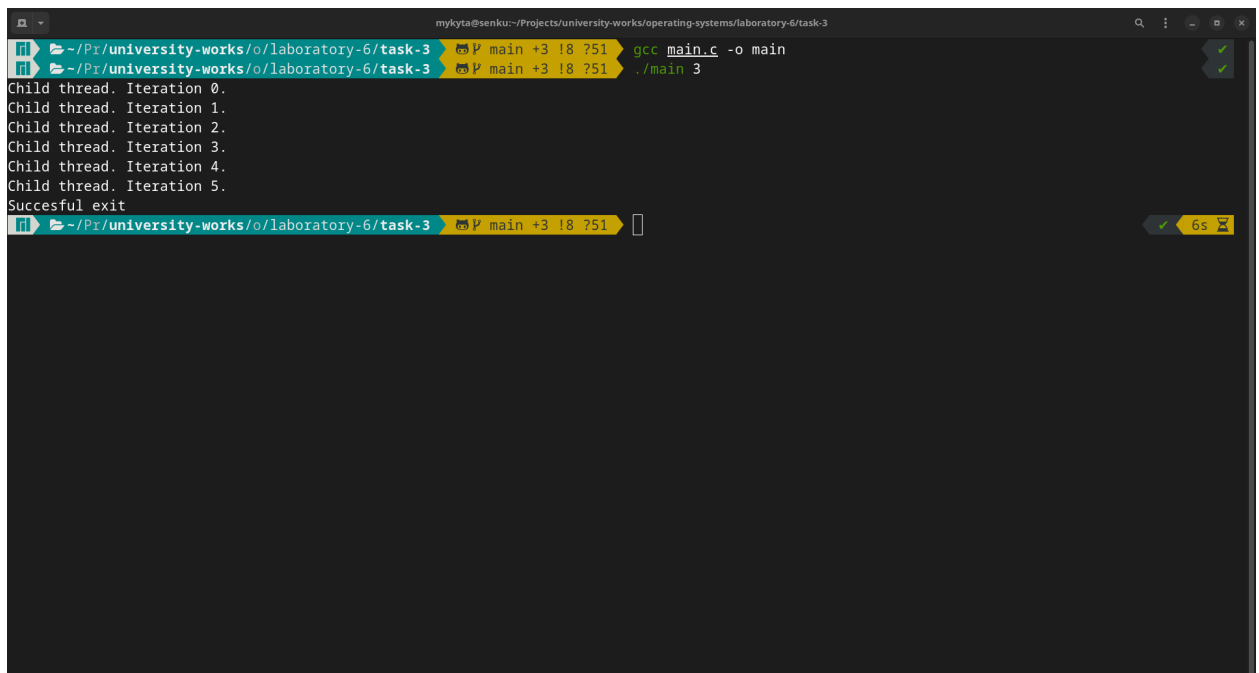


```
mykyta@senku:~/Projects/university-works/operating-systems/laboratory-6/task-2
~/Pr/university-works/o/laboratory-6/task-2 P main +3 !8 ?51 gcc main.c -o main
~/Pr/university-works/o/laboratory-6/task-2 P main +3 !8 ?51 ./main 3
Child thread. Iteration 0.
Child thread. Iteration 1.
Child thread. Iteration 2.
Canceled
~/Pr/university-works/o/laboratory-6/task-2 P main +3 !8 ?51 [ ] 3s
```

## Завдання 3

Напишіть програму, яка за допомогою командного рядка отримує параметр – час затримки в секундах. У основній програмі створюється потік – нащадок і очікується завершення його роботи. Потік-нащадок встановлюється в такий стан, що не дозволяє його скасувати. Цей потік виконання має виводити на екран рядок з текстом (цей рядок включає номер ітерації) задану кількість разів (це дорівнює подвоєному часу затримки) і «засинає» на секунду після кожного виведення. Після створення потоку основний потік засипає на задану кількість секунд (час затримки) і після пробудження намагається скасувати потік-нащадок, що працює. Потім аналізується статус завершення потоку і виводиться повідомлення про те, як завершився потік-нащадок - в результаті скасування або звичайним чином.

## Відповідь



```

mykyta@senku:~/Projects/university-works/operating-systems/laboratory-6/task-3
~/Pr/university-works/o/laboratory-6/task-3 main +3 18 ?51 gcc main.c -o main
~/Pr/university-works/o/laboratory-6/task-3 main +3 18 ?51 ./main 3
Child thread. Iteration 0.
Child thread. Iteration 1.
Child thread. Iteration 2.
Child thread. Iteration 3.
Child thread. Iteration 4.
Child thread. Iteration 5.
Successful exit
~/Pr/university-works/o/laboratory-6/task-3 main +3 18 ?51 6s

```

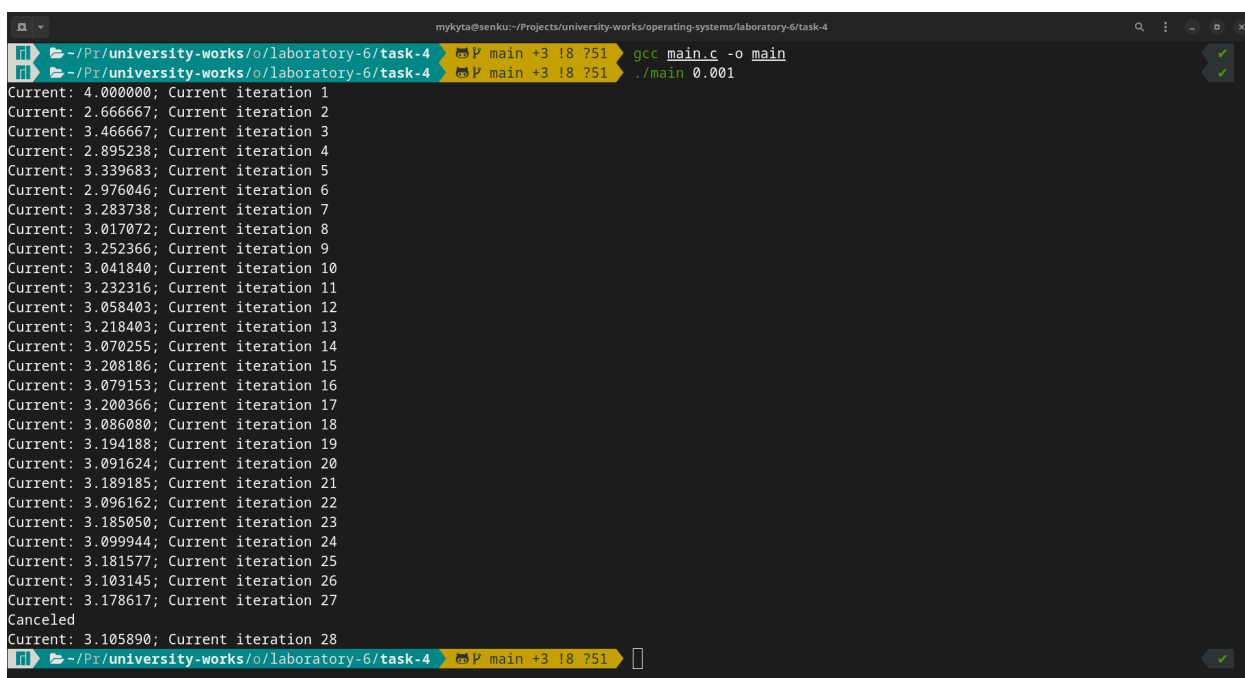
## Завдання 4

Напишіть програму, яка за допомогою командного рядка отримує параметр – час затримки в секундах. У основній програмі створюється потік – нащадок і очікується завершення його роботи. При створенні даний потік – нащадок встановлюється в режим потоку, що синхронно скасовується. У цьому потоці обчислюється значення числа  $\pi$  за формулою Лейбніца:

Обчислення продовжуються до досягнення заданого номеру доданку  $n$  ( $n=100000$ ). Поточне значення числа  $\pi$  виводиться в стандартний потік виведення. Обчислення чергового доданку і додавання його в суму відбувається в секції коду, що не скасовується. Відразу після виходу із цієї секції формується можлива точка виходу. Основний потік виконання очікує на заданий при запуску програми час (кількість секунд), та після очікування посилає команду на скасування створеного

потіку. Потім виконується аналіз: за рахунок чого було завершено потік. Якщо створений потік виконання було скасовано, то вивести відповідне повідомлення. Якщо потік не був скасований, а дорахував задачу до кінця, то вивести обчислений результат.

## Відповідь



```

mykyta@senku:~/Projects/university-works/operating-systems/laboratory-6/task-4
~/Pr/university-works/o/laboratory-6/task-4 main +3 18 751 gcc main.c -o main
~/Pr/university-works/o/laboratory-6/task-4 main +3 18 751 ./main 0.001
Current: 4.000000; Current iteration 1
Current: 2.666667; Current iteration 2
Current: 3.466667; Current iteration 3
Current: 2.895238; Current iteration 4
Current: 3.339683; Current iteration 5
Current: 2.976046; Current iteration 6
Current: 3.283738; Current iteration 7
Current: 3.017072; Current iteration 8
Current: 3.252366; Current iteration 9
Current: 3.041840; Current iteration 10
Current: 3.232316; Current iteration 11
Current: 3.058403; Current iteration 12
Current: 3.218403; Current iteration 13
Current: 3.070255; Current iteration 14
Current: 3.208186; Current iteration 15
Current: 3.079153; Current iteration 16
Current: 3.200366; Current iteration 17
Current: 3.086080; Current iteration 18
Current: 3.194188; Current iteration 19
Current: 3.091624; Current iteration 20
Current: 3.189185; Current iteration 21
Current: 3.096162; Current iteration 22
Current: 3.185050; Current iteration 23
Current: 3.099944; Current iteration 24
Current: 3.181577; Current iteration 25
Current: 3.103145; Current iteration 26
Current: 3.178617; Current iteration 27
Canceled
Current: 3.105890; Current iteration 28

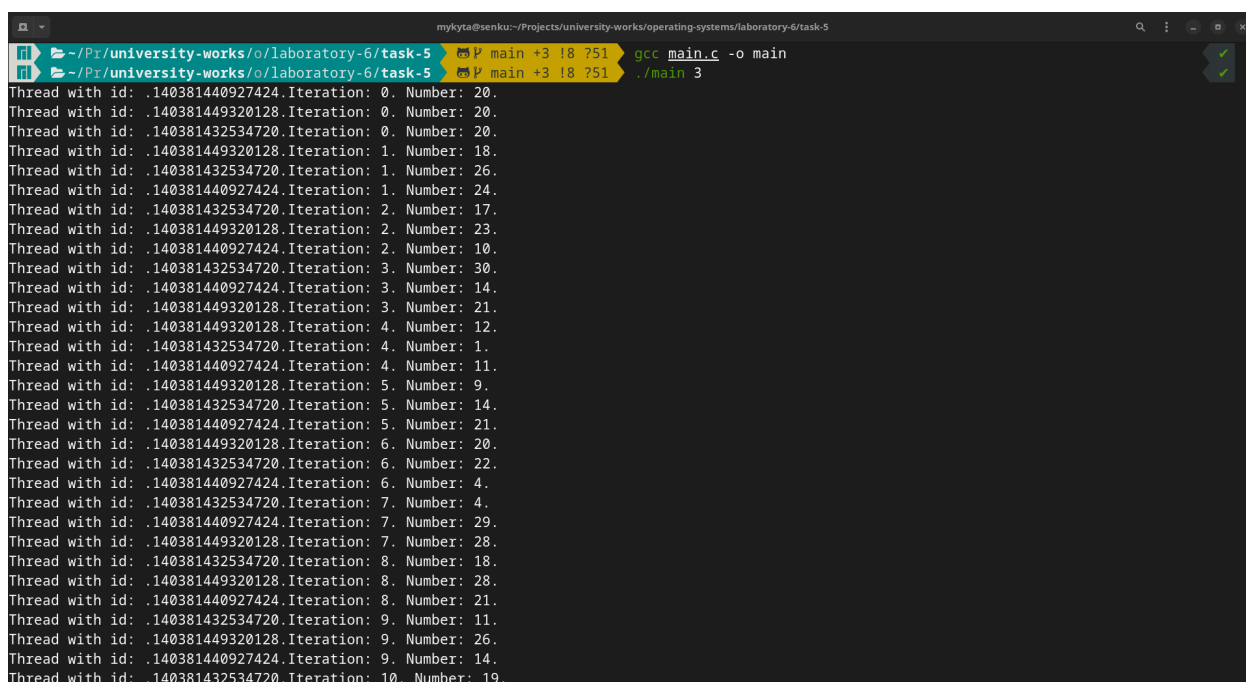
```

## Завдання 5

Напишіть програму, яка за допомогою командного рядка отримує параметр — кількість потоків-нащадків, що будуть створені. В основній програмі (для цього можна створити окрему функцію) створюється задана при старті програми кількість потоків - нащадків та очікується завершення їхньої роботи. Кожен потік виконання виводить у стандартний потік виведення випадкову кількість рядків, що містять випадкові числа. Випадкові числа генеруються із заданого діапазону. Кожен рядок, що виводиться в кожному потоку виконання, включає ідентифікатор потоку,

текстове повідомлення та згенероване псевдовипадкове число. Незмінна частина рядка зберігається в області потокових даних кожного потоку. Можна використати функцію `sleep`, щоб зробити роботу програми більш наочною.

## Відповідь



```
mykyta@senku:~/Projects/university-works/operating-systems/laboratory-6/task-5
~/Px/university-works/o/laboratory-6/task-5 main +3 18 751 gcc main.c -o main
~/Px/university-works/o/laboratory-6/task-5 main +3 18 751 ./main 3
Thread with id: .140381440927424.Iteration: 0. Number: 20.
Thread with id: .140381449320128.Iteration: 0. Number: 20.
Thread with id: .140381432534720.Iteration: 0. Number: 20.
Thread with id: .140381449320128.Iteration: 1. Number: 18.
Thread with id: .140381432534720.Iteration: 1. Number: 26.
Thread with id: .140381440927424.Iteration: 1. Number: 24.
Thread with id: .140381432534720.Iteration: 2. Number: 17.
Thread with id: .140381449320128.Iteration: 2. Number: 23.
Thread with id: .140381440927424.Iteration: 2. Number: 10.
Thread with id: .140381432534720.Iteration: 3. Number: 30.
Thread with id: .140381440927424.Iteration: 3. Number: 14.
Thread with id: .140381449320128.Iteration: 3. Number: 21.
Thread with id: .140381449320128.Iteration: 4. Number: 12.
Thread with id: .140381432534720.Iteration: 4. Number: 1.
Thread with id: .140381440927424.Iteration: 4. Number: 11.
Thread with id: .140381449320128.Iteration: 5. Number: 9.
Thread with id: .140381432534720.Iteration: 5. Number: 14.
Thread with id: .140381440927424.Iteration: 5. Number: 21.
Thread with id: .140381449320128.Iteration: 6. Number: 20.
Thread with id: .140381432534720.Iteration: 6. Number: 22.
Thread with id: .140381440927424.Iteration: 6. Number: 4.
Thread with id: .140381432534720.Iteration: 7. Number: 4.
Thread with id: .140381440927424.Iteration: 7. Number: 29.
Thread with id: .140381449320128.Iteration: 7. Number: 28.
Thread with id: .140381432534720.Iteration: 8. Number: 18.
Thread with id: .140381449320128.Iteration: 8. Number: 28.
Thread with id: .140381440927424.Iteration: 8. Number: 21.
Thread with id: .140381432534720.Iteration: 9. Number: 11.
Thread with id: .140381449320128.Iteration: 9. Number: 26.
Thread with id: .140381440927424.Iteration: 9. Number: 14.
Thread with id: .140381432534720.Iteration: 10. Number: 19.
```

## Завдання 6

Модифікуйте програму із Завдання No2 так, щоб потік-нащадок перед своїм завершенням виводив повідомлення про це з даними, отриманими з нього, у стандартний потік виведення помилок. Для того, щоб гарантувати виконання заданої дії перед завершенням потоку виконання використовуйте обробник очищення, що встановлюється за допомогою функцій `pthread_cleanup_push` та `pthread_cleanup_pop`.

```

mykyta@senku:~/Projects/university-works/operating-systems/laboratory-6/task-6
~/Pr/university-works/o/laboratory-6/task-6 main +3 18 751 ./main 3
Child thread. Iteration 0.
Child thread. Iteration 1.
Child thread. Iteration 2.
pthread_cleanup_push processed.
Canceled
~/Pr/university-works/o/laboratory-6/task-6 3s

```

```

mykyta@senku:~/Projects/university-works/operating-systems/laboratory-6/task-5
Thread with id: .140381440927424.Iteration: 7. Number: 29.
Thread with id: .140381449320128.Iteration: 7. Number: 28.
Thread with id: .140381432534720.Iteration: 8. Number: 18.
Thread with id: .140381449320128.Iteration: 8. Number: 28.
Thread with id: .140381440927424.Iteration: 8. Number: 21.
Thread with id: .140381432534720.Iteration: 9. Number: 11.
Thread with id: .140381449320128.Iteration: 9. Number: 26.
Thread with id: .140381440927424.Iteration: 9. Number: 14.
Thread with id: .140381432534720.Iteration: 10. Number: 19.
Thread with id: .140381440927424.Iteration: 10. Number: 21.
Thread with id: .140381449320128.Iteration: 10. Number: 1.
Thread with id: .140381432534720.Iteration: 11. Number: 8.
Thread with id: .140381449320128.Iteration: 11. Number: 30.
Thread with id: .140381440927424.Iteration: 11. Number: 18.
Thread with id: .140381432534720.Iteration: 12. Number: 1.
Thread with id: .140381449320128.Iteration: 12. Number: 8.
Thread with id: .140381440927424.Iteration: 12. Number: 2.
Thread with id: .140381440927424.Iteration: 13. Number: 3.
Thread with id: .140381432534720.Iteration: 13. Number: 7.
Thread with id: .140381449320128.Iteration: 13. Number: 15.
Thread with id: .140381440927424.Iteration: 14. Number: 15.
Thread with id: .140381432534720.Iteration: 14. Number: 7.
Thread with id: .140381449320128.Iteration: 14. Number: 17.
Thread with id: .140381432534720.Iteration: 15. Number: 8.
Thread with id: .140381449320128.Iteration: 15. Number: 22.
Thread with id: .140381440927424.Iteration: 15. Number: 26.
Thread with id: .140381449320128.Iteration: 16. Number: 8.
Thread with id: .140381432534720.Iteration: 16. Number: 27.
Thread with id: .140381440927424.Iteration: 16. Number: 13.
Thread 0 finished successfully.
Thread 1 finished successfully.
Thread 2 finished successfully.
~/Pr/university-works/o/laboratory-6/task-5 17s

```