Лабораторная работа №6  
Соколов Арсений

**Задание**

Написать программу на С++, где каждый толстяк в своём потоке жрёт наггетсы со своей тарелки. Повар в своём потоке эти наггетсы готовит и выкладывает на стол.

**Дано**

* изначально есть 3 тарелки по 3000 наггетсов в каждой
* если наггетсы закончатся хотя бы у одного толстяка - повара придется уволить.
* скушав больше 10 000 наггетсов толстяк самоуничтожается
* если лопнут все 3 толстяка, Кук не получит зарплату
* по истечению 5 дней Кук увольняется (1 день = 1 секунда компа)
* *gluttony* - столько наггетсов съедает каждый толстяк за раз.
* *efficiency factor* - столько наггетсов Кук выкладывает за раз на стол, раскладывая по 3 тарелкам сразу
* когда Кук выкладывает наггетсы на стол, толстяки ждут

**Найти**

При каких коэффициентах прожорливости толстяков и производительности Кука выполняться условия? (вывести в терминал):

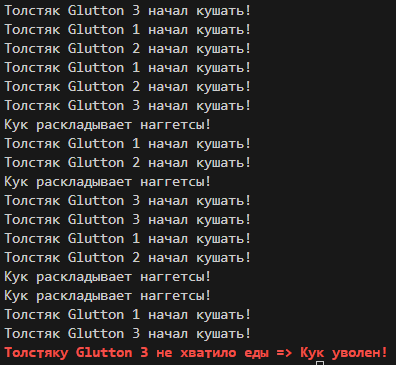
1. Кука уволили
2. Кук не получил зарплату.
3. Кук уволился сам!

**Решение**

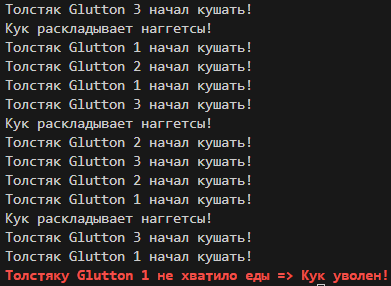
1. **Кука уволили**

***gluttony* > 10** и ***efficiency factor* < 2**

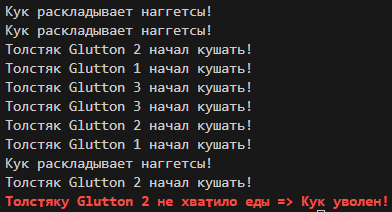
Запуск 1

****

Запуск 2

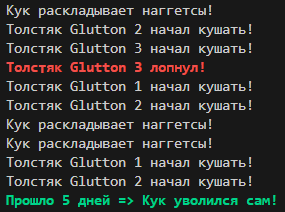
****

Запуск 3



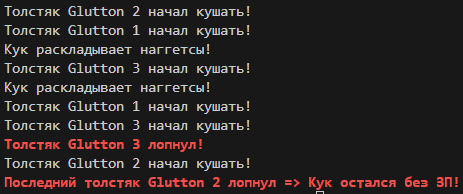
1. **Кук не получил зарплату**

Интересный запуск (*gluttony* = 33 и *efficiency factor = 25*)

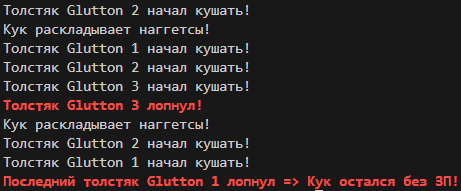


***gluttony* > 34** и ***efficiency factor > 25***

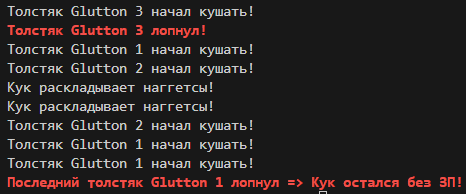
Запуск 1



Запуск 2



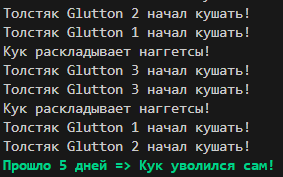
Запуск 3



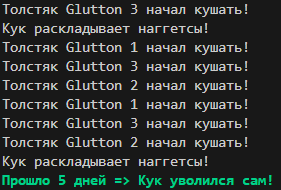
**Кук уволился сам!**

***gluttony* = 1** и ***efficiency factor* = 1**

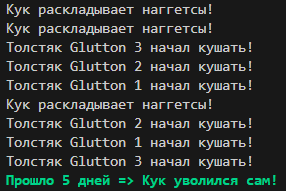
Запуск 1



Запуск 2



Запуск 3



**ПРИЛОЖЕНИЕ 1**

***Программа «ТРИ ТОЛСТЯКА»***

#include <iostream>

#include <thread>

#include <mutex>

#include <chrono>

class Table {

public:

    Table(int dish1, int dish2, int dish3) : dish1(dish1), dish2(dish2), dish3(dish3) {}

    // тут Кук раскладывает наггетсы

    void Cook(int efficiencyFactor, std::chrono::time\_point<std::chrono::high\_resolution\_clock> start) {

        while (true) {

            m.lock();

            if ((std::chrono::high\_resolution\_clock::now() - start) >= std::chrono::seconds(5))

            {

                std::cout << "\033[1;32mПрошло 5 дней => Кук уволился сам!\033[0m" << std::endl;

                exit(1); // конец программы

            }

            std::cout << "Кук раскладывает наггетсы!" << std::endl;

            dish1 += efficiencyFactor;

            dish2 += efficiencyFactor;

            dish3 += efficiencyFactor;

            m.unlock();

            std::this\_thread::sleep\_for(std::chrono::milliseconds(1)); // искусственно создаю возможность начать работу другому потоку

        }

    }

    // тут толстяк кушает наггетсы

    void Glutton(int gluttony, const std::string& name, std::chrono::time\_point<std::chrono::high\_resolution\_clock> start) {

        int eaten = 0; // персональный счетчик съеденных наггетсов

        while (true) {

            if ((std::chrono::high\_resolution\_clock::now() - start) >= std::chrono::seconds(5)) // скорее бы прошло 5 дней

            {

                m.lock();

                std::cout << "\033[1;32mПрошло 5 дней => Кук уволился сам!\033[0m" << std::endl;

                exit(1); // конец программы

            }

            m.lock();

            std::cout << "Толстяк " << name << " начал кушать!" << std::endl;

            if (name == "Glutton 1" && dish1 >= gluttony) { // смотри какой из толстяков ест и хватает ли ему еды в тарелке

                dish1 -= gluttony; // кушает

                eaten += gluttony; // считаем для этого толстяка

                totalEaten += gluttony; // считаем для всех толстяков

                m.unlock();

            } else if (name == "Glutton 2" && dish2 >= gluttony) {

                dish2 -= gluttony;

                eaten += gluttony;

                totalEaten += gluttony;

                m.unlock();

            } else if (name == "Glutton 3" && dish3 >= gluttony) {

                dish3 -= gluttony;

                eaten += gluttony;

                totalEaten += gluttony;

                m.unlock();

            } else {

                std::cout << "\033[1;31mТолстяку " << name << " не хватило еды => Кук уволен!\033[0m" << std::endl;

                exit(1); // конец программы

            }

            if (totalEaten >= death \* 3)

            {

                m.lock();

                std::cout << "\033[1;31mПоследний толстяк " << name << " лопнул => Кук остался без ЗП!\033[0m" << std::endl;

                exit(1); // конец программы

            }

            if (eaten >= death)

            {

                m.lock();

                std::cout << "\033[1;31mТолстяк " << name << " лопнул!\033[0m" << std::endl;

                m.unlock();

                return;

            }

            std::this\_thread::sleep\_for(std::chrono::milliseconds(1)); // искусственно создаю возможность начать работу другому потоку

        }

    }

private:

    int dish1; // тарелка 1

    int dish2; // тарелка 2

    int dish3; // тарелка 3

    int totalEaten = 0; // всего съедено

    int death = 10000; // предел толстяка

    std::mutex m;

};

int main() {

    int gluttony = 1; // коэффициент прожорливости

    int efficiencyFactor = 1; // коэффициент производительности

    auto start = std::chrono::high\_resolution\_clock::now(); // засекаем время начала работы Кука

    Table table(3000, 3000, 3000); // заполняем тарелки

    std::thread cookThread(&Table::Cook, &table, efficiencyFactor, start);

    std::thread glutton1Thread(&Table::Glutton, &table, gluttony, "Glutton 1", start);

    std::thread glutton2Thread(&Table::Glutton, &table, gluttony, "Glutton 2", start);

    std::thread glutton3Thread(&Table::Glutton, &table, gluttony, "Glutton 3", start);

    cookThread.join();

    glutton1Thread.join();

    glutton2Thread.join();

    glutton3Thread.join();

    return 0;

}