ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа бакалавриата «Программная инженерия»

МИКРОПРОЕКТ ПО ДИСЦИПЛИНЕ «АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ»
ПО МНОГОПОТОЧНОМУ ПРОГРАММИРОВАНИЮ НА C++
Пояснительная записка

Исполнитель
Студентка группы БПИ193(подгруппа 1)
Шевко Марина Николаевна
Вариант 4
13 декабря 2020 г

Москва 2020

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

1.1. Постановка задачи на выполнение

Условие (Вариант 4):

Задача об обедающих философах. Пять философов сидят возле круглого стола. Они проводят жизнь, чередуя приемы пищи и размышления. В центре стола находится большое блюдо спагетти. Спагетти длинные и запутанные, философам тяжело управляться с ними, поэтому каждый из них, чтобы съесть порцию, должен пользоваться двумя вилками. К несчастью, философам дали только пять вилок. Между каждой парой философов лежит одна вилка, поэтому эти высококультурные и предельно вежливые люди договорились, что каждый будет пользоваться только теми вилками, которые лежат рядом с ним (слева и справа). Написать многопоточную программу, моделирующую поведение философов с помощью семафоров. Программа должна избегать фатальной ситуации, в которой все философы голодны, но ни один из них не может взять обе вилки (например, каждый из философов держит по одной вилки и не хочет отдавать ее). Решение должно быть симметричным, то есть все потоки-философы должны выполнять один и тот же код.

2. ОПИСАНИЕ ИСПОЛЬЗУЕМЫХ АЛГОРИТМОВ

2.1. Анализ задачи и выбор алгоритма

Задача об обедающих философах является классической задачей, придуманной Э. Дейкстрой, используемой для иллюстрации проблем синхронизации при разработке программ, а именно – deadlock (взаимоблокировка).

Пусть есть 5 потоков, симулирующие действия философов. Для того, чтобы начать есть философ должен взять 2 вилки – слева и справа от него. Т. к. по условию решение должно быть *симметричным*, пусть голодный философ берет сначала левую вилку, а затем правую. Тут же возникает взаимоблокировка потоков – если каждый из философов возьмет вилку слева от него, то никто не сможет взять правую и будут ждать вечно.

Существует решение, предложенное самим Дейкстрой – “Иерархия ресурсов”. Оно заключается в том, что вилки нумеруются и философ всегда берет в левую руку вилку с меньшим номером. Однако, тогда действия философов получаются несимметричными, т.к. замыкающий круг философ, будет выполнять другое действие.

Именно поэтому был выбран способ решения с официантом. Введем дополнительную сущность – официанта, который будет представлен мьютексом. Каждый из философов может начать есть только если свободны обе вилки, по обе стороны от него, то есть, если оба его соседа не едят. Чтобы избежать проблем с использованием старой информации, помогать философам будет официант, который видит ситуацию за столом и может разрешить философу взять вилки, а может попросить его подождать до тех пор, пока его соседи не освободят вилки. Для отслеживания ситуации за столом у каждого философа есть 3 состояния – он размышляет, голоден и ест.

Таким образом, алгоритм следующий:

1. Все философы изначально размышляют какое-то время.
2. Когда философ закончил размышлять, он просит у официанта разрешения на то, чтобы взять 2 вилки и изменяет свое состояние на голоден.
   Если оба его соседа не едят, то он берет две вилки и его состояние изменяется на ест. Иначе, официант предлагает философу подождать, пока оба его соседа не закончат трапезу. Философ остается в состоянии голода.
3. Когда философ закончил есть, он снова возвращается в состояние размышлений. Официант же разрешает взять вилки тем философам - соседям, кто находится в состоянии голода и теперь имеет возможность взять обе вилки.

2.2. Используемые средства

Для работы с потоками используется стандартная библиотека C++ <thread>. Для создания семафора были использованы mutex и condition_variable.

Для симуляции каждого философа используется поток. За каждым философом закреплен семафор. Для симуляции официанта используется мьютекс.

```
class Semaphore {
public:
    void notify() {
        unique_lock<mutex> lock(mtx);
        count++;
        cv.notify_one();
    }

    void wait() {
        unique_lock<mutex> lock(mtx);
        while (count == 0) {
            cv.wait(lock);
        }
        count--;
    }

private:
    mutex mtx;
    condition_variable cv;
    int count = 0;
};
```

2.3. Комментарии к коду

1. Создаем 5 философов – 5 потоков. Время выполнения программы – 15 секунд + время на завершение цикла.

```
cout << endl << "-------------------------Waiter-Report----------------------------" << endl;
t_start = std::chrono::high_resolution_clock::now();
t_end = t_start + 15s;

// Создание потоков и запуск метода обеда
thread philosofers[5];
for (int i = 0; i < 5; i++)
{
    philosofers[i] = thread(dinning, i);
}
```

2. Метод обеда. Каждый философ выполняет одинаковую последовательность действий. Думает, запрашивает вилки у официанта, ест и возвращает вилки на место.

```
void dinning(int num)
{
    while (t_end > chrono::high_resolution_clock::now())
    {
        thinking();               // философ занят раздумьями
        printInfo(num, " asks waiter for forks");
        askForForks(num);         // философ просит официанта дать ему вилки. Если хотя бы одна из вилок занята
                                  // официант просит философа подождать, пока его соседи не закончат есть.
        eating();                 // философ ест
        printInfo(num, " put forks");
        putForks(num);            // философ заканчивает есть. Освободившиеся вилки официант разрешает взять философам, ждущим своей очереди.
    }
}
```

3. Метод запроса вилок. Философ просит вилки у официанта. Официант (мьютекс) блокирует доступ к состояниям философов изменяет его на голоден. После этого официант проверяет доступность вилок.

```
/// <summary>
/// Официант проверяет наличие свободных вилок. Если они есть, философ начинает есть, если их нет
/// - остается в статусе "голодный" и ждет, когда его сосед доест.
/// </summary>
/// <param name="phnum"></param>
void askForForks(int num)
{
    waiter.lock();
    philoStates[num] = "hungry";
    printInfo(num, " is hungry");
    checkAvailability(num);
    waiter.unlock();
    philo[num].wait(); // если свободные вилки были, то значение семафора уменьшится до 0 и он ждать не будет.
                       // если вилка была занята, то философ будет ждать до тех пор, пока официант не проверит, что обе вилки освободились.
}
```

3.1. Проверка доступности вилок. Если философ справа и слева имеют состояние – ест, значит текущий голодный философ не начнет есть. Значение в семафоре останется равным 0 и при вызове метода wait() философ начнет ожидание. Если же проверка пройдена, значение переменной в семафоре становится равным 1. Затем в методе askForForks() при вызове wait() философ ждать не будет.

```
/// <summary>
/// Официант проверяет свободны ли обе вилки вокруг философа. Если да, он разрешает философу взять вилки и он ест.
/// Если хотя бы одна из вилок занята (сосед философа ест), то философ остается со статусом "голоден"
/// </summary>
/// <param name="phnum"></param>
void checkAvailability(int num)
{
    if (philoStates[num] == "hungry" && philoStates[(num + 4) % 5] != "eating" && philoStates[(num + 1) % 5] != "eating")
    {
        philoStates[num] = "eating";
        printInfo(num, " is eating");
        philo[num].notify();
    }
}
```

4. Метод опускания вилок. Официант опять блокирует состояния философов и изменяет его статус на размышляет. Теперь нам доступны 2 вилки и официант проверяет соседей – слева и справа не голодны ли они и могут ли они теперь начать трапезу. Если да, то в методе значение семафора снова увеличивается до 1, метод wait() заканчивает выполнение и философ начинает есть.

```
/// <summary>
/// Философ кладет вилки на стол и погружается в размышления.
/// Официант проверяет, есть ли справа и слева от закончившего трапезу философа те, кто голоден и ждет вилки.
/// Если такой есть, и при этом вилка с другой стороны не занята, то вилки отдаются ему.
/// </summary>
/// <param name="phnum"></param>
void putForks(int num)
{
    waiter.lock();
    philoStates[num] = "thinking";
    printInfo(num, " is thinking");
    checkAvailability((num + 4) % 5);
    checkAvailability((num + 1) % 5);
    waiter.unlock();
}
```

3. ТЕСТИРОВНИЕ ПРОГРАММЫ

1. Изначально все философы размышляют (разное время в зависимости от рандомного числа). Аристотель первый захотел есть и просит вилки у официанта. Так как никакие вилки пока не заняты, то он начинает есть. Затем есть хочет Эпикур и так как он находится через одного человека, то ему тоже дают вилки и он начинает есть. Затем захотел есть Платон, но та как он сидит между Аристотелем и Эпикуром, официант ему вилки не даст, так как они заняты. И он останется ждать в состоянии голода. То же самое с Парменидом и Сократом. Затем Эпикур закончил есть, положил вилки и перешел в состояние раздумий. Сразу же после этого Платон, который был соседом Эпикура и ждал вилки их получил и начал есть. И так далее. Выполнение заканчивается тем, что все положили вилки и погрузились в раздумья.

```
Five philosofers are sitting at the table in the following sequence:
0-Aristotle 1-Parmenides 2-Plato 3-Epicurus 4-Socrates
-------------------------Waiter-Report----------------------------
 0-Aristotle is thinking
 1-Parmenides is thinking
 2-Plato is thinking
 3-Epicurus is thinking
 4-Socrates is thinking
0-Aristotle asks waiter for forks
   0-Aristotle is hungry
       0-Aristotle is eating
3-Epicurus asks waiter for forks
   3-Epicurus is hungry
       3-Epicurus is eating
2-Plato asks waiter for forks
   2-Plato is hungry
1-Parmenides asks waiter for forks
   1-Parmenides is hungry
4-Socrates asks waiter for forks
   4-Socrates is hungry
          3-Epicurus put forks
 3-Epicurus is thinking
       2-Plato is eating
          0-Aristotle put forks
 0-Aristotle is thinking
       4-Socrates is eating
          4-Socrates put forks
 4-Socrates is thinking
          2-Plato put forks
 2-Plato is thinking
0-Aristotle asks waiter for forks
3-Epicurus asks waiter for forks
       1-Parmenides is eating
   0-Aristotle is hungry
   3-Epicurus is hungry
       3-Epicurus is eating
4-Socrates asks waiter for forks
   4-Socrates is hungry
2-Plato asks waiter for forks
   2-Plato is hungry
          1-Parmenides put forks
 1-Parmenides is thinking
       0-Aristotle is eating
          3-Epicurus put forks
 3-Epicurus is thinking
       2-Plato is eating
          2-Plato put forks
 2-Plato is thinking
          0-Aristotle put forks
 0-Aristotle is thinking
       4-Socrates is eating
1-Parmenides asks waiter for forks
   1-Parmenides is hungry
       1-Parmenides is eating
3-Epicurus asks waiter for forks
   3-Epicurus is hungry
2-Plato asks waiter for forks
   2-Plato is hungry
0-Aristotle asks waiter for forks
   0-Aristotle is hungry
          4-Socrates put forks
 4-Socrates is thinking
       3-Epicurus is eating
          1-Parmenides put forks
 1-Parmenides is thinking
       0-Aristotle is eating
1-Parmenides asks waiter for forks
   1-Parmenides is hungry
4-Socrates asks waiter for forks
   4-Socrates is hungry
          3-Epicurus put forks
 3-Epicurus is thinking
       2-Plato is eating
          0-Aristotle put forks
 0-Aristotle is thinking
       4-Socrates is eating
          2-Plato put forks
 2-Plato is thinking
       1-Parmenides is eating
          1-Parmenides put forks
 1-Parmenides is thinking
          4-Socrates put forks
 4-Socrates is thinking
```

2. Тут все разом попросили у официанта вилки, но выдали их только через одного Аристотелю и Платону – у них оба соседа не едят, соответственно, вилки свободны. Остальные Парменид, Сократ и Эпикур остались в состоянии голода ждать. Затем Платон (2) кладет вилки и Эпикур (3) его сосед начинаете есть. И так далее.

```
Five philosofers are sitting at the table in the following sequence:
0-Aristotle 1-Parmenides 2-Plato 3-Epicurus 4-Socrates
-------------------------Waiter-Report----------------------------
 0-Aristotle is thinking
 1-Parmenides is thinking
 2-Plato is thinking
 3-Epicurus is thinking
 4-Socrates is thinking
0-Aristotle asks waiter for forks
1-Parmenides asks waiter for forks
2-Plato asks waiter for forks
3-Epicurus asks waiter for forks
4-Socrates asks waiter for forks
   0-Aristotle is hungry
       0-Aristotle is eating
   1-Parmenides is hungry
   2-Plato is hungry
       2-Plato is eating
   3-Epicurus is hungry
   4-Socrates is hungry
          2-Plato put forks
 2-Plato is thinking
       3-Epicurus is eating
          0-Aristotle put forks
 0-Aristotle is thinking
       1-Parmenides is eating
2-Plato asks waiter for forks
0-Aristotle asks waiter for forks
   2-Plato is hungry
   0-Aristotle is hungry
          3-Epicurus put forks
 3-Epicurus is thinking
       4-Socrates is eating
3-Epicurus asks waiter for forks
   3-Epicurus is hungry
          4-Socrates put forks
 4-Socrates is thinking
       3-Epicurus is eating
4-Socrates asks waiter for forks
   4-Socrates is hungry
          1-Parmenides put forks
 1-Parmenides is thinking
       0-Aristotle is eating
1-Parmenides asks waiter for forks
   1-Parmenides is hungry
          3-Epicurus put forks
 3-Epicurus is thinking
       2-Plato is eating
          0-Aristotle put forks
 0-Aristotle is thinking
       4-Socrates is eating
3-Epicurus asks waiter for forks
   3-Epicurus is hungry
0-Aristotle asks waiter for forks
   0-Aristotle is hungry
          2-Plato put forks
 2-Plato is thinking
       1-Parmenides is eating
          4-Socrates put forks
 4-Socrates is thinking
       3-Epicurus is eating
2-Plato asks waiter for forks
   2-Plato is hungry
4-Socrates asks waiter for forks
          3-Epicurus put forks
          1-Parmenides put forks
   4-Socrates is hungry
 3-Epicurus is thinking
       4-Socrates is eating
3-Epicurus asks waiter for forks
 1-Parmenides is thinking
       2-Plato is eating
          4-Socrates put forks
```

4. СПИСОК ЛИТЕРАТУРЫ

1) Задача об обедающих философах [Электронный ресурс] // Режим доступа: Задача об обедающих философах — Википедия (wikipedia.org), свободный (Дата обращения: 12.12.2020)

2) Semaphore C++ 11 [Электронный ресурс] // Режим доступа: https://riptutorial.com/cplusplus/example/30142/semaphore-cplusplus-11 , свободный (Дата обращения: 12.12.2020)

3) Проблема обедающих философов [Электронный ресурс] // Режим доступа: https://ru.qaz.wiki/wiki/Dining_philosophers_problem , свободный (Дата обращения: 12.12.2020)

4) Solutions to the dining philosopher problem [Электронный ресурс] // Режим доступа: https://www.stolaf.edu/people/rab/pdc/text/dpsolns.htm , свободный (Дата обращения: 12.12.2020)

5) Modern dining philosophers [Электронный ресурс] // Режим доступа: http://lucteo.ro/2018/12/28/modern-dining-philosophers/ , свободный (Дата обращения: 12.12.2020)