ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
Факультет компьютерных наук
Департамент программной инженерии

ЗАДАЧА О ВИННИ ПУХЕ (вариант 2)

Отчёт

Дисциплина: «Архитектура вычислительных систем»

Исполнитель:
студент группы БПИ195
Баранова Екатерина

Москва 2020

СОДЕРЖАНИЕ

1. ТЕКСТ ЗАДАНИЯ

В одном лесу живут n пчел и один медведь, которые используют один горшок меда, вместимостью H глотков. Сначала горшок пустой. Пока горшок не наполнится, медведь спит. Как только горшок заполняется, медведь просыпается и съедает весь мед, после чего снова засыпает. Каждая пчела многократно собирает по одному глотку меда и кладет его в горшок. Пчела, которая приносит последнюю порцию меда, будит медведя. Создать многопоточное приложение, моделирующее поведение пчел и медведя.

2. ПРИМЕНЯЕМЫЕ РАСЧЕТНЫЕ МЕТОДЫ

Исполнение программы начинается с ввода двух параметров через командную консоль. Первый параметр – количество пчел, второй – количество глотков. После этого создаются потоки в указанных количествах. В каждом потоке запускается 20 секундный таймер, который ограничивает время работы программы.

В методе для пчел они приносят мед и увеличивают количество глотков, которые может сделать медведь, а мьютексы блокируют вывод, благодаря им все выводится в правильном порядке.

Медведь же спит до наполнения меда, после просыпается и обнуляет количество меда, после чего засыпает.

3. ТЕСТИРОВАНИЕ ПРОГРАММЫ

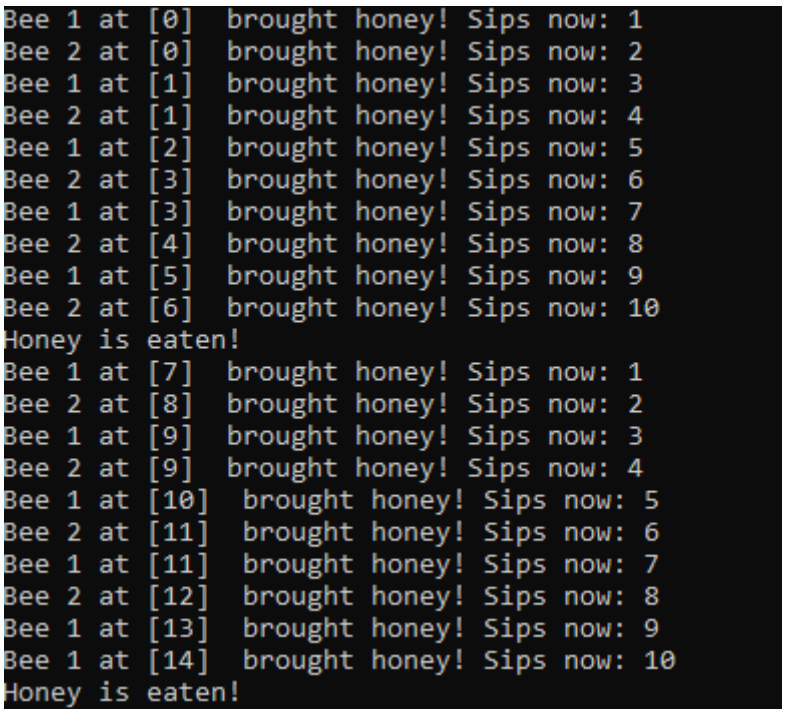

Рисунок 1 – Программа работает корректно при 2 пчелах и 10 глотках, потоки чередуются, вывод оформлен правильно

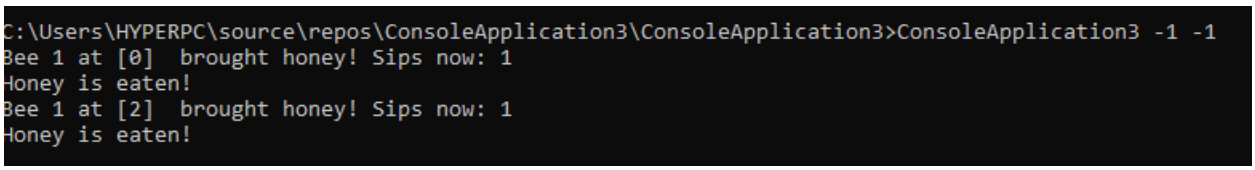

Рисунок 2 – При некорректных данных программа установит значения 1, 1 по умолчанию

ИСТОЧНИКИ

1. SoftCraft, сайт по учебной дисциплине. [Электронный ресурс] http://softcraft.ru/ (дата обращения: 12.12.2020).

ПРИЛОЖЕНИЕ 1

КОД ПРОГРАММЫ

```
#include <iostream>
#define HAVE_STRUCT_TIMESPEC
#include <pthread.h>
#include <semaphore.h>
#include <windows.h>
#include <chrono>
#include <string>
#include <vector>
using namespace std;
#pragma comment(lib,"pthreadVC2.lib")

int beeNumber;
int totalSips;
int sipNumber;
pthread_mutex_t allowEat;

void* Bee(void* args)
{
	//Начало и конец по времени
	auto start = chrono::system_clock::now();
	auto end = std::chrono::system_clock::now();
	//Номер пчелы
	int num = *((int*)args);
	srand(time(0) + num);
	int delay = 1000 + (rand() % 10) * 100;
	while ((std::chrono::duration_cast<std::chrono::seconds>(end - start).count() <=
20)) {
		//Уменьшаем количество входных потоков семафора
		pthread_mutex_lock(&allowEat);
		if (sipNumber < totalSips)
		{
			//Блокируем вывод, чтобы текст не бегал
			cout << "Bee " << num << " at [" <<
chrono::duration_cast<chrono::seconds>(chrono::system_clock::now() - start).count() << "]
"  << " brought honey! ";
			sipNumber++;
			cout << "Sips now: " << sipNumber << endl;

			//Разблокируем обратно
		}
		pthread_mutex_unlock(&allowEat);
		//Пчела впадает в спячку
		Sleep(delay);
		end = std::chrono::system_clock::now();
	}
	return NULL;
}

void* Bear(void* args)
{
	//Медведь спит столько же по времени
	auto start = chrono::system_clock::now();
	auto end = std::chrono::system_clock::now();
	int sipAmount = *((int*)args);
	while ((std::chrono::duration_cast<std::chrono::seconds>(end - start).count() <=
20)) {
		//Проходим по семафору и ждем пока не наберется меда
		if (sipNumber == totalSips)
		{
			//Мишка ест мед
```

```
                        sipNumber = 0;
                        pthread_mutex_lock(&allowEat);
                        cout << "Honey is eaten!" << endl;
                        pthread_mutex_unlock(&allowEat);
                }
                Sleep(1500);
                end = std::chrono::system_clock::now();
        }
        return NULL;
}

int main(int argc, char* argv[])
{
        //Проверка входных аргументов
        if (argc != 3)
        {
                cout << "Incorrect amount of parametrs, please enter amount of bees and
food amount" << endl;
                return -1;
        }
        try {
                totalSips = stoi(argv[1]);
                if (totalSips <= 0) { totalSips = 1; }
                sipNumber = 0;
                beeNumber = stoi(argv[2]);
                if (beeNumber <= 0) { beeNumber = 1; }
        }
        catch (exception e) {
                cout << "Incorrect data";
                return -1;
        }
        //Инициализация мутексов и семафоров
        pthread_mutex_init(&allowEat, nullptr);
        int* arr = new int[beeNumber];
        vector<pthread_t> bees(beeNumber);
        pthread_t bear;
        //Создаем медведя
        pthread_create(&bear, NULL, Bear, &totalSips);
        //Создаем пчел
        for (int t = 0; t < beeNumber; t++)
        {
                arr[t] = t + 1;
                pthread_create(&bees[t], NULL, Bee, &arr[t]);
        }
        //После завершения работы соединяем все потоки
        for (int t = 0; t < beeNumber; t++)
        {
                pthread_join(bees[t], NULL);
        }
        delete[] arr;
        pthread_join(bear, NULL);
}
```