Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет «Высшая школа экономики»

Факультет компьютерных наук

Отчет к домашней работе №4.1

По дисциплине

“Архитектура вычислительных систем”

Работу выполнил

Студент группы БПИ-197 _____ А. А. Синенко

подпись, дата

Работу проверил _____ А. И. Легалов

подпись, дата

Москва 2020

Содержание

Постановка задачи

В цехе по заточке булавок все необходимые операции осуществляются тремя рабочими. Первый из них берет булавку и проверяет ее на предмет кривизны. Если булавка не кривая, то рабочий передает ее своему напарнику. Напарник осуществляет собственно заточку и передает заточенную булавку третьему рабочему, который осуществляет контроль качества операции. Требуется создать многопоточное приложение, моделирующее работу цеха. При решении использовать парадигму «производитель-потребитель».

Уточнение задачи

1) Необходимо при помощи библиотеки OpenMP по работе с потоками в C++ написать программу, моделирующую работу цеха по производству булавок (pin).
2) Выводить действия рабочих в консоль.
3) Использовать парадигму “Производитель-потребитель” (1, стр. 1).

Реализация программы

- Для хранения информации о производимой булавке создана структура pin.
- Для моделирования конвейеров созданы три очереди (queue) с булавками `conveer1, conveer2, conveer3.`
- Поле `producted` для хранения произведенных деталей.
- Поле `defective` для хранения произведенных деталей.
- Методы-потоки `firstWorker, secondWorker, thirdWorker` для моделирования работы трех рабочих.
- В методе `int main()` реализовано создание трех потоков, ожидание завершения этих потоков и корректное завершение программы.

Тестирование программы

При корректном запуске программа выводит информацию о любом действии рабочих индивидуально.

```
Worker C: Остался недоволен работой рабочего B. Возвращает деталь 86 рабочему B.
Worker B: Заточил деталь 83 и передал ее рабочему C.
Worker C: Закончил производство детали 93.
Worker B: Заточил деталь 96 и передал ее рабочему C.
Worker C: Закончил производство детали 87.
Worker B: Заточил деталь 97 и передал ее рабочему C.
Worker C: Остался недоволен работой рабочего B. Возвращает деталь 88 рабочему B.
Worker C: Остался недоволен работой рабочего B. Возвращает деталь 83 рабочему B.
Worker B: Заточил деталь 98 и передал ее рабочему C.
Worker C: Остался недоволен работой рабочего B. Возвращает деталь 96 рабочему B.
Worker B: Заточил деталь 86 и передал ее рабочему C.
Worker C: Закончил производство детали 97.
Worker C: Закончил производство детали 98.
Worker B: Заточил деталь 88 и передал ее рабочему C.
Worker B: Заточил деталь 83 и передал ее рабочему C.
Worker C: Закончил производство детали 86.
Worker B: Заточил деталь 96 и передал ее рабочему C.
Worker C: Закончил производство детали 88.
Worker C: Закончил производство детали 83.
Worker C: Остался недоволен работой рабочего B. Возвращает деталь 96 рабочему B.
Worker C: Остался недоволен работой рабочего B. Возвращает деталь 96 рабочему B.
Worker B: Заточил деталь 96 и передал ее рабочему C.
Worker B: Заточил деталь 96 и передал ее рабочему C.
Worker C: Остался недоволен работой рабочего B. Возвращает деталь 96 рабочему B.
Worker B: Заточил деталь 96 и передал ее рабочему C.
Worker C: Закончил производство детали 96.
Worker C: done
Worker B: done
Всего произведено: 52
Всего брака: 48
```

Рисунок – демонстрация работы программы (последние строки).

Использованная литература

1. https://studydocs.ru/studfiles/112/916/656121/9.doc.html (электронный ресурс)
2. https://docs.microsoft.com/ru-ru/cpp/parallel/openmp/reference/openmp-library-reference?view=msvc-160 (электронный ресурс)

Приложение

```
#include <iostream>
#include <queue>
#include <omp.h>
using namespace std;

const size_t pinsCount = 100;
//const size_t pinsCount = 1000;
//const size_t pinsCount = 10000;
// Прогресс рабочих (0 - все работают, 1 - A закончил работу, 2 - C и B закончили работу)
int status = 0;

// Класс детали
struct pin
{
	// Идентификатор детали
	int id = 0;
	// Является ли деталь кривой
	bool isCurved = false;
	// Качественная ли заточка детали
	bool isGoodQuality = false;

	pin(int id)
	{
		isCurved = rand() % 2;
		this->id = id;
	}
};

// Рабочий A (производитель)
void first_worker(queue<pin>& input, queue<pin>& output, queue<pin>& defective)
{
	while (!input.empty())
	{
		// Берется деталь
		pin takenPin = input.front();
		input.pop();
		if (!takenPin.isCurved)
		{
			// Передается рабочему B
			output.push(takenPin);
			#pragma omp critical
			{
				cout << "Worker A: Проверил деталь " << takenPin.id << " и передал рабочему B" << endl;
			}
		}
		else
		{
			// Бракуется
			defective.push(takenPin);
			#pragma omp critical
			{
				cout << "Worker A: Деталь " << takenPin.id << " оказалась кривой." << endl;
```

```
56                      }
57              }
58
59              status = 0;
60      }
61      cout << "Worker A: done" << endl;
62      status = 1;
63 }
64
65 // Рабочий B (потребитель для A и производитель для C)
66 void second_worker(queue<pin>& input, queue<pin>& output)
67 {
68      while (true)
69      {
70              while (!input.empty())
71              {
72                      // Берется деталь
73                      pin takenPin = input.front();
74                      input.pop();
75                      // Заточка и передача рабочему C
76                      takenPin.isGoodQuality = rand() % 2;
77                      output.push(takenPin);
78                      #pragma omp critical
79                      {
80                              cout << "Worker B: Заточил деталь " << takenPin.id
81 << " и передал ее рабочему C." << endl;
82                      }
83              }
84              if (status == 2)
85                      break;
86      }
87      cout << "Worker B: done" << endl;
88 }
89
90 // Рабочий C (потребитель)
91 void third_worker(queue<pin>& input, queue<pin>& secondWorkerInput, queue<pin>&
92 output)
93 {
94      while (true)
95      {
96              while (!input.empty())
97              {
98                      // Берется деталь
99                      pin takenPin = input.front();
100                     input.pop();
101                     if (takenPin.isGoodQuality)
102                     {
103                             // Заканчивает производство детали
104                             output.push(takenPin);
105                             #pragma omp critical
106                             {
107                                     cout << "Worker C: Закончил производство
108 детали " << takenPin.id << "." << endl;
109                             }
110                             // Условие окончания работы цеха
111                             if (input.empty() && secondWorkerInput.empty() &&
112 status == 1)
113                                     status = 2;
```

```
114                         }
115                         else
116                         {
117                                 // Возвращается рабочему B на доработку
118                                 secondWorkerInput.push(takenPin);
119                                 #pragma omp critical
120                                 {
121                                         cout << "Worker C: Остался недоволен работой
122 рабочего B. "
123                                                 << "Возвращает деталь " <<
124 takenPin.id << " рабочему B." << endl;
125                                 }
126                         }
127                 }
128                 if (status == 2)
129                         break;
130         }
131         cout << "Worker C: done" << endl;
132 }
133
134 // Входная точка программы
135 int main()
136 {
137         setlocale(LC_ALL, "Russian");
138         if (pinsCount == 0)
139         {
140                 cout << "Всего произведено: 0" << endl;
141                 cout << "Всего брака: 0" << endl;
142                 return 0;
143         }
144         // Обнуление состояния
145         status = 0;
146         // Вход для рабочего A
147         queue<pin> conveer1;
148         for (int i = 0; i < pinsCount; i++)
149                 conveer1.emplace(pin(i + 1));
150         // Вход для рабочего B
151         queue<pin> conveer2;
152         // Вход для рабочего C
153         queue<pin> conveer3;
154         // Произведенные детали
155         queue<pin> producted;
156         // Забракованные детали
157         queue<pin> defective;
158
159         // Параллелизм по секциям в три потока
160         #pragma omp parallel sections num_threads(3)
161         {
162                 #pragma omp section
163                 {
164                         first_worker(conveer1, conveer2, defective);
165                 }
166                 #pragma omp section
167                 {
168                         second_worker(conveer2, conveer3);
169                 }
170                 #pragma omp section
171                 {
```

```
                third_worker(conveer3, conveer2, producted);
            }
        }
        // Итог
        cout << "Всего произведено: " << producted.size() << endl;
        cout << "Всего брака: " << defective.size() << endl;
        // Выход из программы.
        return 0;
}
```