Opis rozwiązania zadania 4

Mikołaj Olejnik

Zadanie rozwiązałem implementując program w C++. Stworzyłem klasę Buffer, która dziedziczy po klasie Monitor, z dwoma warunkami empty i full, oraz licznikiem elementów w buforze.

```
class Buffer : public Monitor {
    protected:
        string item_name;
        int count;
        int buffer_size;
        Condition empty;
        Condition full;
```

W tej klasie zaimplementowałem dwie metody put i get. Obie metody zwracają aktualną ilość elementów w buforze.

```
int put() {
    enter();

    if (count >= this->buffer_size)
        wait(full);
    count++;
    int return_count = count;
    if (count == 1)
        signal(empty);

    leave();
    return return_count;
}
```

```
int get() {
    enter();

    if (count <= 0)
        wait(empty);
    count--;
    int return_count = count;
    if (count == this->buffer_size - 1)
        signal(full);

    leave();
    return return_count;
}
```

Następnie stworzyłem klasę która reprezentuje pierogarnię. Składa się ona z 4 bufferów, każdy z nich odpowiadający innemu składnikowi.

```
class Pierogarnia {
    public:
        Buffer ciasto;
        Buffer mieso;
        Buffer kapusta;
        Buffer ser;
```

W pliku main.cpp znajduje się główna implementacja mojego rozwiązania. Program przyjmuje 3 argumenty: liczbę producentów, liczbę konsumentów oraz wielkość bufora.

Na początku tworzę producentów, każdy z nich jest osobnym wątkiem. Argumentami funkcji produce jest index producenta oraz buffer do którego będzie on produkował składniki.

```
for (int i = 0; i < producers; i++) {
    args.buffer = buffers[i];
    args.index = indexes[i];
    args_list_producers[i] = args;
    pthread_create(&th_producers[i], NULL, produce, (void *)&args_list_producers[i]);
    sleep(1);
}
```

W tablicy buffers dla każdego producenta umieściłem odpowiadający mu buffer. Na początku upewniłem się, że każdy ze składników jest produkowany przynajmniej przez jednego producenta. Reszta producentów dostała losowy buffer do uzupełniania.

```
void fill_buffers(Pierogarnia *pierogarnia, Buffer* buffers[], int producers)
{
    // we must have all the ingredients for a pierog
    buffers[0] = &pierogarnia->ciasto;
    buffers[1] = &pierogarnia->mieso;
    buffers[2] = &pierogarnia->ser;
    buffers[3] = &pierogarnia->kapusta;

    // if we have more availabe producers, then we want more "ciasto" produced
    if (producers > 4) {
        buffers[4] = &pierogarnia->ciasto;
    }

    // any more available producers will be random
    for (int i=5; i<producers; i++) {
        buffers[i] = get_random_buffer(pierogarnia);
    }
}
```

W funkcji produce w nieskończonej pętli producent tworzy składnik.

```
void *produce(void *arguments)
{
    arg_struct *args = (arg_struct *)arguments;
    Buffer *buffer = args->buffer;
    int index = args->index;
    int buffer_size = args->buffer_size;

    while (true) {
        sleep(1);
        int count = buffer->put();

        printf("Producer %d: Created %s (%d/%d)\n",
               index, buffer->get_item_name().c_str(), count, buffer_size);
    }
}
```

Następnie tworzę konsumentów, którzy będą robili pierogi. Argumentami funkcji consume jest index konsumenta oraz odpowiedni buffer z nadzieniem (ser, kapusta lub mięso).

```
for (int i = 0; i < consumers; i++) {
    args.buffer = buffers_nadzienie[i];
    args.index = indexes[i];
    args_list_producers[i] = args;
    pthread_create(&th_consumers[i], NULL, consume, (void *)&args_list_producers[i]);
    sleep(1);
}
```

W funkcji consume tworzone są pierogi. Każdy z konsumentów ma dostęp do bufora na ciasto i bufora z losowym składnikiem. Jeśli zarówno ciasto jak i nadzienie jest dostępne, to konsument tworzy pieroga.

```
void *consume(void *arguments)
{
    arg_struct *args = (arg_struct *)arguments;
    Buffer *buffer_nadzienie = args->buffer;
    Buffer *buffer_ciasto = args->buffer_ciasto;
    int index = args->index;
    int buffer_size = args->buffer_size;

    while (true) {
        sleep(1);

        int count_nadzienie = buffer_nadzienie->get();
        int count_ciasto = buffer_ciasto->get();

        printf("Consumer %d: Made pierog with %s (%d/%d) (Ciasto %d/%d)\n",
               index, buffer_nadzienie->get_item_name().c_str(), count_nadzienie,
               buffer_size, count_ciasto, buffer_size);
    }
}
```