

# SOI zad. 3 - testowanie poprawności

Testowanie poprawności polega na uruchomieniu programu w tylu konfiguracjach wątków (ilości wątków danego typu), że wyczerpane zostaną wszystkie możliwe scenariusze dostępu do bufora.

## Testy z kolejką początkowo pustą

### 1 wątek A1

Gdy w kolejce znajdzie się 10 elementów, to warunek produkcji wątku A1, przestanie być spełniony, więc nie wyjdzie on z sekcji krytycznej, lecz odda ją innemu wątkowi - który nie istnieje, czyli nigdy tej sekcji krytycznej nie odzyska i wystąpi zakleszczenie.

```
### PUT (A1 0) -> 1[0,] ###
### PUT (A1 2) -> 2[0,2,] ###
### PUT (A1 4) -> 3[0,2,4,] ###
### PUT (A1 6) -> 4[0,2,4,6,] ###
### PUT (A1 8) -> 5[0,2,4,6,8,] ###
### PUT (A1 10) -> 6[0,2,4,6,8,10,] ###
### PUT (A1 12) -> 7[0,2,4,6,8,10,12,] ###
### PUT (A1 14) -> 8[0,2,4,6,8,10,12,14,] ###
### PUT (A1 16) -> 9[0,2,4,6,8,10,12,14,16,] ###
### PUT (A1 18) -> 10[0,2,4,6,8,10,12,14,16,18,] ###
```

### 1 wątek A2

Na samym początku warunek produkcji A2 nie będzie spełniony i wystąpi zakleszczenie

```
olek@olek-PC:~/studia/soi/ex3$ g++ test.cpp -lpthread -o test2
olek@olek-PC:~/studia/soi/ex3$ ./test2
```

### dowolna ilość B1 i B2 , bez żadnego A1 lub A2

W każdym przypadku efekt jest ten sam, wątki konsumujące B1 i B2 nie mogą zjadać liczb gdy bufor jest pusty, czyli wystąpi zakleszczenie.

```
olek@olek-PC:~/studia/soi/ex3$ g++ test.cpp -lpthread -o test2
olek@olek-PC:~/studia/soi/ex3$ ./test2
```

### 1 wątek A1 , 1 A2

Wystąpi zakleszczenie gdy warunki `A1` i `A2` przestaną być spełnione, czyli gdy w kolejce będzie 20 elementów.

```
### PUT (A1 0) -> 1[0,] ###
### PUT (A2 1) -> 2[0,1,] ###
### PUT (A1 2) -> 3[0,1,2,] ###
### PUT (A2 3) -> 4[0,1,2,3,] ###
### PUT (A1 4) -> 5[0,1,2,3,4,] ###
### PUT (A2 5) -> 6[0,1,2,3,4,5,] ###
### PUT (A1 6) -> 7[0,1,2,3,4,5,6,] ###
### PUT (A2 7) -> 8[0,1,2,3,4,5,6,7,] ###
### PUT (A1 8) -> 9[0,1,2,3,4,5,6,7,8,] ###
### PUT (A2 9) -> 10[0,1,2,3,4,5,6,7,8,9,] ###
### PUT (A1 10) -> 11[0,1,2,3,4,5,6,7,8,9,10,] ###
### PUT (A2 11) -> 12[0,1,2,3,4,5,6,7,8,9,10,11,] ###
### PUT (A1 12) -> 13[0,1,2,3,4,5,6,7,8,9,10,11,12,] ###
### PUT (A2 13) -> 14[0,1,2,3,4,5,6,7,8,9,10,11,12,13,] ###
### PUT (A1 14) -> 15[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,] ###
### PUT (A2 15) -> 16[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,] ###
### PUT (A1 16) -> 17[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,] ###
### PUT (A2 17) -> 18[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,] ###
### PUT (A1 18) -> 19[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,] ###
### PUT (A2 19) -> 20[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,] ###
```

## 1 wątek `A1`, 1 `B1`

Wątek `A1` produkuje parzyste, a `B1` konsumuje parzyste, więc zakleszczenie nie wystąpi i program będzie się wykonywał poprawnie.

```
### GET (B1) -> 9[40,42,44,46,48,0,2,4,6,] ###
### PUT (A1 8) -> 10[40,42,44,46,48,0,2,4,6,8,] ###
### GET (B1) -> 9[42,44,46,48,0,2,4,6,8,] ###
### PUT (A1 10) -> 10[42,44,46,48,0,2,4,6,8,10,] ###
### GET (B1) -> 9[44,46,48,0,2,4,6,8,10,] ###
### PUT (A1 12) -> 10[44,46,48,0,2,4,6,8,10,12,] ###
### GET (B1) -> 9[46,48,0,2,4,6,8,10,12,] ###
### PUT (A1 14) -> 10[46,48,0,2,4,6,8,10,12,14,] ###
### GET (B1) -> 9[48,0,2,4,6,8,10,12,14,] ###
### PUT (A1 16) -> 10[48,0,2,4,6,8,10,12,14,16,] ###
### GET (B1) -> 9[0,2,4,6,8,10,12,14,16,] ###
### PUT (A1 18) -> 10[0,2,4,6,8,10,12,14,16,18,] ###
### GET (B1) -> 9[2,4,6,8,10,12,14,16,18,] ###
### PUT (A1 20) -> 10[2,4,6,8,10,12,14,16,18,20,] ###
### GET (B1) -> 9[4,6,8,10,12,14,16,18,20,] ###
### PUT (A1 22) -> 10[4,6,8,10,12,14,16,18,20,22,] ###
### GET (B1) -> 9[6,8,10,12,14,16,18,20,22,] ###
### PUT (A1 24) -> 10[6,8,10,12,14,16,18,20,22,24,] ###
### GET (B1) -> 9[8,10,12,14,16,18,20,22,24,] ###
### PUT (A1 26) -> 10[8,10,12,14,16,18,20,22,24,26,] ###
### GET (B1) -> 9[10,12,14,16,18,20,22,24,26,] ###
### PUT (A1 28) -> 10[10,12,14,16,18,20,22,24,26,28,] ###
### GET (B1) -> 9[12,14,16,18,20,22,24,26,28,] ###
### PUT (A1 30) -> 10[12,14,16,18,20,22,24,26,28,30,] ###
```

## 1 wątek `A2`, 1 `B2`

Mimo, że `A2` produkuje nieparzyste, a `B2` je konsumuje, to na samym początku wystąpi zakleszczenie z tego powodu, że warunek `A2` na starcie nie jest spełniony.

```
olek@olek-PC:~/studia/soi/ex3$ g++ test.cpp -lpthread -o test2
olek@olek-PC:~/studia/soi/ex3$ ./test2
```

## 1 wątek `A1`, 1 `A2`, 1 `B1`, 1 `B2`

Program wykonuje się poprawnie bez zakleszczeń i nieprawidłowych dostępu do bufora.

```
### PUT (A1 36) -> 18[7,18,9,20,11,22,24,13,26,15,28,17,30,19,32,34,21,36,] ###
### PUT (A2 23) -> 19[7,18,9,20,11,22,24,13,26,15,28,17,30,19,32,34,21,36,23,] ###
### GET (B2) -> 18[18,9,20,11,22,24,13,26,15,28,17,30,19,32,34,21,36,23,] ###
### GET (B1) -> 17[9,20,11,22,24,13,26,15,28,17,30,19,32,34,21,36,23,] ###
### PUT (A1 38) -> 18[9,20,11,22,24,13,26,15,28,17,30,19,32,34,21,36,23,38,] ###
### PUT (A2 25) -> 19[9,20,11,22,24,13,26,15,28,17,30,19,32,34,21,36,23,38,25,] ###
### GET (B2) -> 18[20,11,22,24,13,26,15,28,17,30,19,32,34,21,36,23,38,25,] ###
### PUT (A2 27) -> 19[20,11,22,24,13,26,15,28,17,30,19,32,34,21,36,23,38,25,27,] ###
### GET (B1) -> 18[11,22,24,13,26,15,28,17,30,19,32,34,21,36,23,38,25,27,] ###
### PUT (A1 40) -> 19[11,22,24,13,26,15,28,17,30,19,32,34,21,36,23,38,25,27,40,] ###
### GET (B2) -> 18[22,24,13,26,15,28,17,30,19,32,34,21,36,23,38,25,27,40,] ###
### PUT (A2 29) -> 19[22,24,13,26,15,28,17,30,19,32,34,21,36,23,38,25,27,40,29,] ###
### GET (B1) -> 18[24,13,26,15,28,17,30,19,32,34,21,36,23,38,25,27,40,29,] ###
### PUT (A1 42) -> 19[24,13,26,15,28,17,30,19,32,34,21,36,23,38,25,27,40,29,42,] ###
### GET (B1) -> 18[13,26,15,28,17,30,19,32,34,21,36,23,38,25,27,40,29,42,] ###
### GET (B2) -> 17[26,15,28,17,30,19,32,34,21,36,23,38,25,27,40,29,42,] ###
### PUT (A1 44) -> 18[26,15,28,17,30,19,32,34,21,36,23,38,25,27,40,29,42,44,] ###
### GET (B1) -> 17[15,28,17,30,19,32,34,21,36,23,38,25,27,40,29,42,44,] ###
### PUT (A2 31) -> 18[15,28,17,30,19,32,34,21,36,23,38,25,27,40,29,42,44,31,] ###
### GET (B2) -> 17[28,17,30,19,32,34,21,36,23,38,25,27,40,29,42,44,31,] ###
### PUT (A1 46) -> 18[28,17,30,19,32,34,21,36,23,38,25,27,40,29,42,44,31,46,] ###
### GET (B1) -> 17[17,30,19,32,34,21,36,23,38,25,27,40,29,42,44,31,46,] ###
### PUT (A2 33) -> 18[17,30,19,32,34,21,36,23,38,25,27,40,29,42,44,31,46,33,] ###
```

## 2 wątki A1 , 2 A2 , 2 B1 , 2 B2

Program wykonuje się poprawnie bez zakleszczeń i nieprawidłowych dostępu do bufora.

Zwiększenie ilości wątków dla obojętnie której klasy nie zakłóca poprawnego działania programu.

```
### GET (B1) -> 18[2,9,40,4,11,29,13,42,6,15,44,17,46,31,8,19,21,48,] ###
### PUT (A1 10) -> 19[2,9,40,4,11,29,13,42,6,15,44,17,46,31,8,19,21,48,10,] ###
### GET (B1) -> 18[9,40,4,11,29,13,42,6,15,44,17,46,31,8,19,21,48,10,] ###
### GET (B2) -> 17[40,4,11,29,13,42,6,15,44,17,46,31,8,19,21,48,10,] ###
### GET (B1) -> 16[4,11,29,13,42,6,15,44,17,46,31,8,19,21,48,10,] ###
### PUT (A1 0) -> 17[4,11,29,13,42,6,15,44,17,46,31,8,19,21,48,10,0,] ###
### GET (B1) -> 16[11,29,13,42,6,15,44,17,46,31,8,19,21,48,10,0,] ###
### PUT (A1 2) -> 17[11,29,13,42,6,15,44,17,46,31,8,19,21,48,10,0,2,] ###
### PUT (A1 12) -> 18[11,29,13,42,6,15,44,17,46,31,8,19,21,48,10,0,2,12,] ###
### PUT (A2 23) -> 19[11,29,13,42,6,15,44,17,46,31,8,19,21,48,10,0,2,12,23,] ###
### PUT (A2 33) -> 20[11,29,13,42,6,15,44,17,46,31,8,19,21,48,10,0,2,12,23,33,] ###
### GET (B2) -> 19[29,13,42,6,15,44,17,46,31,8,19,21,48,10,0,2,12,23,33,] ###
### PUT (A2 35) -> 20[29,13,42,6,15,44,17,46,31,8,19,21,48,10,0,2,12,23,33,35,] ###
### GET (B2) -> 19[13,42,6,15,44,17,46,31,8,19,21,48,10,0,2,12,23,33,35,] ###
### GET (B2) -> 18[42,6,15,44,17,46,31,8,19,21,48,10,0,2,12,23,33,35,] ###
### GET (B1) -> 17[6,15,44,17,46,31,8,19,21,48,10,0,2,12,23,33,35,] ###
### PUT (A1 4) -> 18[6,15,44,17,46,31,8,19,21,48,10,0,2,12,23,33,35,4,] ###
### GET (B1) -> 17[15,44,17,46,31,8,19,21,48,10,0,2,12,23,33,35,4,] ###
### PUT (A1 14) -> 18[15,44,17,46,31,8,19,21,48,10,0,2,12,23,33,35,4,14,] ###
### PUT (A2 25) -> 19[15,44,17,46,31,8,19,21,48,10,0,2,12,23,33,35,4,14,25,] ###
### PUT (A2 37) -> 20[15,44,17,46,31,8,19,21,48,10,0,2,12,23,33,35,4,14,25,37,] ###
### GET (B2) -> 19[44,17,46,31,8,19,21,48,10,0,2,12,23,33,35,4,14,25,37,] ###
```

Poniżej przykład z: 1 A1 , 1 A2 , 10 B1 , 6 B2

```
### GET (B2) -> 7[2,4,5,4,47,6,8,49,] ###
### GET (B1) -> 6[45,4,47,6,8,49,] ###
### PUT (A1 10) -> 7[45,4,47,6,8,49,10,] ###
### GET (B2) -> 6[4,47,6,8,49,10,] ###
### GET (B1) -> 5[47,6,8,49,10,] ###
### PUT (A2 1) -> 6[47,6,8,49,10,1,] ###
### PUT (A1 12) -> 7[47,6,8,49,10,1,12,] ###
### GET (B2) -> 6[6,8,49,10,1,12,] ###
### GET (B1) -> 5[8,49,10,1,12,] ###
### PUT (A2 3) -> 6[8,49,10,1,12,3,] ###
### GET (B1) -> 5[49,10,1,12,3,] ###
### PUT (A1 14) -> 6[49,10,1,12,3,14,] ###
### PUT (A1 16) -> 7[49,10,1,12,3,14,16,] ###
### PUT (A2 5) -> 8[49,10,1,12,3,14,16,5,] ###
### GET (B2) -> 7[10,1,12,3,14,16,5,] ###
### GET (B1) -> 6[1,12,3,14,16,5,] ###
### PUT (A1 18) -> 7[1,12,3,14,16,5,18,] ###
```

## Testy z kolejką początkowo pełną (10 el.)

W zadaniu kolejka nie ma górnego ograniczenia posiadanych elementów, więc na potrzeby tych testów jest 'pełna' wtedy, gdy A1 nie może generować nowych liczb, czyli ma 10 liczb parzystych. Kolejka na początek ma więc następujące wartości [0, 2, 4, 6, 8, 10, 12, 14, 16, 18]

### 1 wątek A1

Warunek A1 na początku nie jest spełniony, czyli nastąpi zakleszczenie.

```
olek@olek-PC:~/studia/soi/ex3$ g++ ./test.cpp -o test -lpthread
olek@olek-PC:~/studia/soi/ex3$ ./test
^C
```

### 1 wątek A2

Tym razem, na samym początku warunek A2 jest spełniony, więc wątek wygeneruje 10 liczb, po czym warunek generowania nie będzie już dłużej spełniony, więc zakleszczy się.

```
olek@olek-PC:~/studia/soi/ex3$ ./test
### PUT (A2 1) -> 11[0,2,4,6,8,10,12,14,16,18,1,] ###
### PUT (A2 3) -> 12[0,2,4,6,8,10,12,14,16,18,1,3,] ###
### PUT (A2 5) -> 13[0,2,4,6,8,10,12,14,16,18,1,3,5,] ###
### PUT (A2 7) -> 14[0,2,4,6,8,10,12,14,16,18,1,3,5,7,] ###
### PUT (A2 9) -> 15[0,2,4,6,8,10,12,14,16,18,1,3,5,7,9,] ###
### PUT (A2 11) -> 16[0,2,4,6,8,10,12,14,16,18,1,3,5,7,9,11,] ###
### PUT (A2 13) -> 17[0,2,4,6,8,10,12,14,16,18,1,3,5,7,9,11,13,] ###
### PUT (A2 15) -> 18[0,2,4,6,8,10,12,14,16,18,1,3,5,7,9,11,13,15,] ###
### PUT (A2 17) -> 19[0,2,4,6,8,10,12,14,16,18,1,3,5,7,9,11,13,15,17,] ###
### PUT (A2 19) -> 20[0,2,4,6,8,10,12,14,16,18,1,3,5,7,9,11,13,15,17,19,] ###
^C
```

### 1 wątek B1

Wątek B1 zjada liczby parzyste do momentu aż zostanie ich  $< 3$ . I wtedy się zakleszczy.

```
olek@olek-PC:~/studia/soi/ex3$ ./test
### GET (B1) -> 9[2,4,6,8,10,12,14,16,18,] ###
### GET (B1) -> 8[4,6,8,10,12,14,16,18,] ###
### GET (B1) -> 7[6,8,10,12,14,16,18,] ###
### GET (B1) -> 6[8,10,12,14,16,18,] ###
### GET (B1) -> 5[10,12,14,16,18,] ###
### GET (B1) -> 4[12,14,16,18,] ###
### GET (B1) -> 3[14,16,18,] ###
### GET (B1) -> 2[16,18,] ###
^C
```

## 1 wątek B2

Warunek wątku od samego początku nie będzie spełniony, ponieważ bufor zawiera same liczby parzyste - wątek B2 zakleszczy się.

```
olek@olek-PC:~/studia/soi/ex3$ g++ ./test.cpp -o test -lpthread
olek@olek-PC:~/studia/soi/ex3$ ./test
^C
```

## 1 wątek A1, 1 A2

Sytuacja taka sama gdy mamy tylko wątek A2, ponieważ warunek A1 będzie od samego początku nie spełniony.

```
olek@olek-PC:~/studia/soi/ex3$ ./test
### PUT (A2 1) -> 11[0,2,4,6,8,10,12,14,16,18,1,] ###
### PUT (A2 3) -> 12[0,2,4,6,8,10,12,14,16,18,1,3,] ###
### PUT (A2 5) -> 13[0,2,4,6,8,10,12,14,16,18,1,3,5,] ###
### PUT (A2 7) -> 14[0,2,4,6,8,10,12,14,16,18,1,3,5,7,] ###
### PUT (A2 9) -> 15[0,2,4,6,8,10,12,14,16,18,1,3,5,7,9,] ###
### PUT (A2 11) -> 16[0,2,4,6,8,10,12,14,16,18,1,3,5,7,9,11,] ###
### PUT (A2 13) -> 17[0,2,4,6,8,10,12,14,16,18,1,3,5,7,9,11,13,] ###
### PUT (A2 15) -> 18[0,2,4,6,8,10,12,14,16,18,1,3,5,7,9,11,13,15,] ###
### PUT (A2 17) -> 19[0,2,4,6,8,10,12,14,16,18,1,3,5,7,9,11,13,15,17,] ###
### PUT (A2 19) -> 20[0,2,4,6,8,10,12,14,16,18,1,3,5,7,9,11,13,15,17,19,] ###
^C
```

## 1 wątek A1, 1 B1

Sytuacja podobna jak w przypadku pustej kolejki przy inicjalizacji, różnica jest taka, że w tym przypadku to wątek B1 uzyska na samym początku dostęp do kolejki, ponieważ warunek dla A1 z początku nie jest spełniony.

```
### PUT (A1 14) -> 10[46,48,0,2,4,6,8,10,12,14,] ###
### GET (B1) -> 9[48,0,2,4,6,8,10,12,14,] ###
### PUT (A1 16) -> 10[48,0,2,4,6,8,10,12,14,16,] ###
### GET (B1) -> 9[0,2,4,6,8,10,12,14,16,] ###
### PUT (A1 18) -> 10[0,2,4,6,8,10,12,14,16,18,] ###
### GET (B1) -> 9[2,4,6,8,10,12,14,16,18,] ###
### PUT (A1 20) -> 10[2,4,6,8,10,12,14,16,18,20,] ###
### GET (B1) -> 9[4,6,8,10,12,14,16,18,20,] ###
### PUT (A1 22) -> 10[4,6,8,10,12,14,16,18,20,22,] ###
### GET (B1) -> 9[6,8,10,12,14,16,18,20,22,] ###
### PUT (A1 24) -> 10[6,8,10,12,14,16,18,20,22,24,] ###
### GET (B1) -> 9[8,10,12,14,16,18,20,22,24,] ###
### PUT (A1 26) -> 10[8,10,12,14,16,18,20,22,24,26,] ###
### GET (B1) -> 9[10,12,14,16,18,20,22,24,26,] ###
### PUT (A1 28) -> 10[10,12,14,16,18,20,22,24,26,28,] ###
```



## 1 wątek A2, 1 B1

Przez to, że A2 produkuje l. nieparzyste, a B1 konsumuje tylko parzyste, to gdy w buforze skończą się liczby parzyste, to wystąpi zakleszczenie.

```
olek@olek-PC:~/studia/soi/ex3$ ./test
### PUT (A2 1) -> 11[0,2,4,6,8,10,12,14,16,18,1,] ###
### GET (B1) -> 10[2,4,6,8,10,12,14,16,18,1,] ###
### PUT (A2 3) -> 11[2,4,6,8,10,12,14,16,18,1,3,] ###
### GET (B1) -> 10[4,6,8,10,12,14,16,18,1,3,] ###
### PUT (A2 5) -> 11[4,6,8,10,12,14,16,18,1,3,5,] ###
### GET (B1) -> 10[6,8,10,12,14,16,18,1,3,5,] ###
### PUT (A2 7) -> 11[6,8,10,12,14,16,18,1,3,5,7,] ###
### GET (B1) -> 10[8,10,12,14,16,18,1,3,5,7,] ###
### PUT (A2 9) -> 11[8,10,12,14,16,18,1,3,5,7,9,] ###
### GET (B1) -> 10[10,12,14,16,18,1,3,5,7,9,] ###
### GET (B1) -> 9[12,14,16,18,1,3,5,7,9,] ###
### GET (B1) -> 8[14,16,18,1,3,5,7,9,] ###
### GET (B1) -> 7[16,18,1,3,5,7,9,] ###
### GET (B1) -> 6[18,1,3,5,7,9,] ###
### GET (B1) -> 5[1,3,5,7,9,] ###
```

## 1 wątek A2, 1 B2

mimo, że A2 produkuje l. nieparzyste, a B2 konsumuje nieparzyste, to i tak wystąpi zakleszczenie, spowodowane tym, że B2 może konsumować tylko z początku kolejki, a tam znajdują się liczby parzyste, którymi została ona zainicjalizowana.

```
olek@olek-PC:~/studia/soi/ex3$ ./test
### PUT (A2 1) -> 11[0,2,4,6,8,10,12,14,16,18,1,] ###
### PUT (A2 3) -> 12[0,2,4,6,8,10,12,14,16,18,1,3,] ###
### PUT (A2 5) -> 13[0,2,4,6,8,10,12,14,16,18,1,3,5,] ###
### PUT (A2 7) -> 14[0,2,4,6,8,10,12,14,16,18,1,3,5,7,] ###
### PUT (A2 9) -> 15[0,2,4,6,8,10,12,14,16,18,1,3,5,7,9,] ###
### PUT (A2 11) -> 16[0,2,4,6,8,10,12,14,16,18,1,3,5,7,9,11,] ###
### PUT (A2 13) -> 17[0,2,4,6,8,10,12,14,16,18,1,3,5,7,9,11,13,] ###
### PUT (A2 15) -> 18[0,2,4,6,8,10,12,14,16,18,1,3,5,7,9,11,13,15,] ###
### PUT (A2 17) -> 19[0,2,4,6,8,10,12,14,16,18,1,3,5,7,9,11,13,15,17,] ###
### PUT (A2 19) -> 20[0,2,4,6,8,10,12,14,16,18,1,3,5,7,9,11,13,15,17,19,] ###
```

## 1 wątek A1, 1 A2, 1 B1, 1 B2

Program wykonuje się poprawnie bez zakleszczeń i nieprawidłowych dostępu do bufora.

```
### GET (B2) -> 18[24,26,39,28,41,30,43,32,45,34,47,36,49,38,40,1,42,3,] ###
### GET (B1) -> 17[26,39,28,41,30,43,32,45,34,47,36,49,38,40,1,42,3,] ###
### PUT (A2 5) -> 18[26,39,28,41,30,43,32,45,34,47,36,49,38,40,1,42,3,5,] ###
### PUT (A1 44) -> 19[26,39,28,41,30,43,32,45,34,47,36,49,38,40,1,42,3,5,44,] ###
### GET (B1) -> 18[39,28,41,30,43,32,45,34,47,36,49,38,40,1,42,3,5,44,] ###
### GET (B2) -> 17[28,41,30,43,32,45,34,47,36,49,38,40,1,42,3,5,44,] ###
### PUT (A2 7) -> 18[28,41,30,43,32,45,34,47,36,49,38,40,1,42,3,5,44,7,] ###
### PUT (A1 46) -> 19[28,41,30,43,32,45,34,47,36,49,38,40,1,42,3,5,44,7,46,] ###
### PUT (A2 9) -> 20[28,41,30,43,32,45,34,47,36,49,38,40,1,42,3,5,44,7,46,9,] ###
### GET (B1) -> 19[41,30,43,32,45,34,47,36,49,38,40,1,42,3,5,44,7,46,9,] ###
### GET (B2) -> 18[30,43,32,45,34,47,36,49,38,40,1,42,3,5,44,7,46,9,] ###
### PUT (A1 48) -> 19[30,43,32,45,34,47,36,49,38,40,1,42,3,5,44,7,46,9,48,] ###
### PUT (A2 11) -> 20[30,43,32,45,34,47,36,49,38,40,1,42,3,5,44,7,46,9,48,11,] ###
### GET (B1) -> 19[43,32,45,34,47,36,49,38,40,1,42,3,5,44,7,46,9,48,11,] ###
### GET (B2) -> 18[32,45,34,47,36,49,38,40,1,42,3,5,44,7,46,9,48,11,] ###
### PUT (A1 0) -> 19[32,45,34,47,36,49,38,40,1,42,3,5,44,7,46,9,48,11,0,] ###
### PUT (A2 13) -> 20[32,45,34,47,36,49,38,40,1,42,3,5,44,7,46,9,48,11,0,13,] ###
### GET (B1) -> 19[45,34,47,36,49,38,40,1,42,3,5,44,7,46,9,48,11,0,13,] ###
### GET (B2) -> 18[34,47,36,49,38,40,1,42,3,5,44,7,46,9,48,11,0,13,] ###
### GET (B1) -> 17[47,36,49,38,40,1,42,3,5,44,7,46,9,48,11,0,13,] ###
### PUT (A1 2) -> 18[47,36,49,38,40,1,42,3,5,44,7,46,9,48,11,0,13,2,] ###
### GET (B2) -> 17[36,49,38,40,1,42,3,5,44,7,46,9,48,11,0,13,2,] ###
```

## 2 wątki A1 , 2 A2 , 2 B1 , 2 B2

Program wykonuje się poprawnie bez zakleszczeń i nieprawidłowych dostępu do bufora.

Zwiększenie ilości wątków dla obojętnie której klasy nie zakłóca poprawnego działania programu.

```
### GET (B2) -> 19[23,40,4,9,6,42,11,44,25,13,8,27,46,10,15,48,17,29,0,] ###
### PUT (A2 19) -> 20[23,40,4,9,6,42,11,44,25,13,8,27,46,10,15,48,17,29,0,19,] ###
### GET (B2) -> 19[40,4,9,6,42,11,44,25,13,8,27,46,10,15,48,17,29,0,19,] ###
### PUT (A2 31) -> 20[40,4,9,6,42,11,44,25,13,8,27,46,10,15,48,17,29,0,19,31,] ###
### GET (B1) -> 19[4,9,6,42,11,44,25,13,8,27,46,10,15,48,17,29,0,19,31,] ###
### PUT (A1 2) -> 20[4,9,6,42,11,44,25,13,8,27,46,10,15,48,17,29,0,19,31,2,] ###
### GET (B1) -> 19[9,6,42,11,44,25,13,8,27,46,10,15,48,17,29,0,19,31,2,] ###
### GET (B2) -> 18[6,42,11,44,25,13,8,27,46,10,15,48,17,29,0,19,31,2,] ###
### PUT (A1 12) -> 19[6,42,11,44,25,13,8,27,46,10,15,48,17,29,0,19,31,2,12,] ###
### PUT (A2 21) -> 20[6,42,11,44,25,13,8,27,46,10,15,48,17,29,0,19,31,2,12,21,] ###
### GET (B1) -> 19[42,11,44,25,13,8,27,46,10,15,48,17,29,0,19,31,2,12,21,] ###
### PUT (A1 4) -> 20[42,11,44,25,13,8,27,46,10,15,48,17,29,0,19,31,2,12,21,4,] ###
### GET (B1) -> 19[11,44,25,13,8,27,46,10,15,48,17,29,0,19,31,2,12,21,4,] ###
### GET (B2) -> 18[44,25,13,8,27,46,10,15,48,17,29,0,19,31,2,12,21,4,] ###
```

Poniżej przykład z: 1 A1 , 1 A2 , 10 B1 , 6 B2

```
### PUT (A1 4) -> 6[47,0,49,2,1,4,] ###
### PUT (A1 6) -> 7[47,0,49,2,1,4,6,] ###
### PUT (A2 3) -> 8[47,0,49,2,1,4,6,3,] ###
### GET (B2) -> 7[0,49,2,1,4,6,3,] ###
### GET (B1) -> 6[49,2,1,4,6,3,] ###
### PUT (A1 8) -> 7[49,2,1,4,6,3,8,] ###
### PUT (A2 5) -> 8[49,2,1,4,6,3,8,5,] ###
### GET (B2) -> 7[2,1,4,6,3,8,5,] ###
### GET (B1) -> 6[1,4,6,3,8,5,] ###
### PUT (A1 10) -> 7[1,4,6,3,8,5,10,] ###
```