Name :- Khan Faisal
Roll no :- 197
Batch :- C12

SAP ID : 60004240019
Subj :- OS

# Experiment . (4)

* Aim :- Implement program on Producer & Consumer problem.

* Theory :-

The producer-consumer problem is a classical synchronization problem in OS. where two processes shares fixed-sized buffer. The producer generates data & adds it to buffer, while the consumer removes & processes it.

• Key challenges :-

1. Synchronization : The producer should not add data if buffer is full. the consumer should not remove data if buffer is empty

2. Race condition : If both processes access the buffer simultaneously, it may lead to inconsistency.

3. Deadlock : Improper synchronization can cause both process to wait indefinitely.

• Solution Approach :

To resolve this problem, we use :

1] Semaphore
2] Mutex
3] Monitors

· Simple Analogy :-

Imagine a toy factory (Producer) & a toy store (consumer) sharing a limited shelf (Buffer).

i] the factory produces toys & places them on the shelf.

ii] The store picks toy from the shelf to sell.

a] If shelf is full, factory must wait until space is limited.

b] If shell is ~~ful~~ empty, factory must wait until new toys arrive.

c] only one person can access the shelf at a time to avoid confusion.

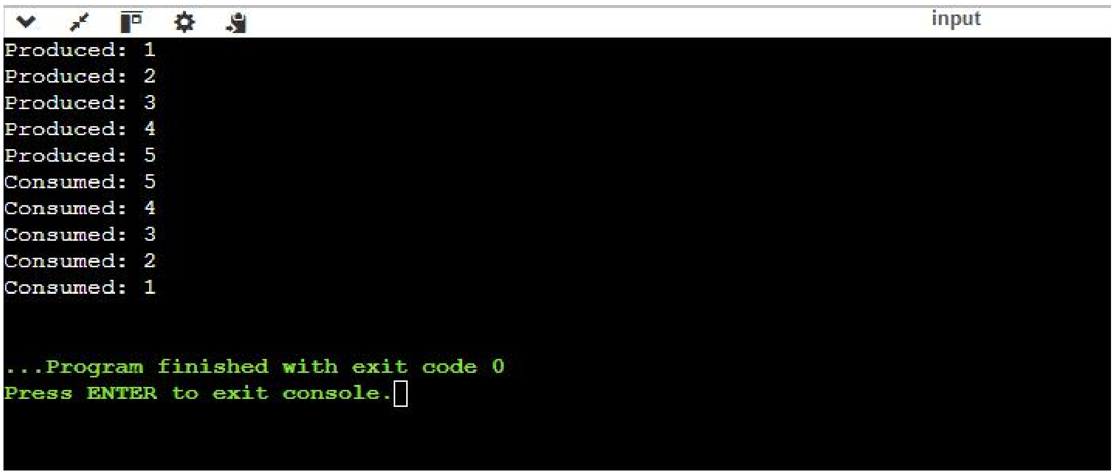
* Conclusion :-

Thus, I have understood the concept of producers-consumer problem. & it's program in c. methods to resolve this class syndronization problem.

**Program (Sequential Execution):**

```
#include <stdio.h>

#define BUFFER_SIZE 5

int buffer[BUFFER_SIZE];
int item_count = 0; // Tracks how many items are in the buffer

// Producer: Adds items to the buffer
void produce(int item) {
    if (item_count >= BUFFER_SIZE) {
        printf("Buffer full! Cannot produce.\n");
        return;
    }
    buffer[item_count++] = item;
    printf("Produced: %d\n", item);
}

// Consumer: Removes and processes items from the buffer
int consume() {
    if (item_count <= 0) {
        printf("Buffer empty! Cannot consume.\n");
        return -1; // Error: No item to consume
    }
    int item = buffer[--item_count];
    printf("Consumed: %d\n", item);
    return item;
}

int main() {
    // Producer adds items
    for (int i = 1; i <= 5; i++) {
        produce(i);
    }
    // Consumer takes items
    while (item_count > 0) {
        consume();
    }

    return 0;
}
```

**Output:**



```
Produced: 1
Produced: 2
Produced: 3
Produced: 4
Produced: 5
Consumed: 5
Consumed: 4
Consumed: 3
Consumed: 2
Consumed: 1


...Program finished with exit code 0
Press ENTER to exit console.
```