

## Producer Consumer OS Lab

*By 1BM21CS232*

```
#include<stdio.h>
#include<stdlib.h>

int mutex=1,full=0,empty=3,x=0;
void producer();
void consumer();
int wait(int);
int signal(int);
int main()
{
    int n;
    printf("\n 1.Producer\n 2.Consumer\n");
    while(1)
    {
        printf("\nEnter your choice");
        scanf("%d",&n);
        switch(n)
        {
            case 1: if((mutex==1)&&(empty==0))
                    printf("Buffer is full");
                    else
                    producer();
                    break;
            case 2: if((mutex==1)&&(full!=0))
                    consumer();
                    else
                    printf("Buffer is empty");
```

```
        break;
    case 3: exit(0);
        break;
    }
}
return 0;
}
```

```
int wait(int s)
{
    return(--s);
}
```

```
int signal(int s)
{
    return(++s);
}
```

```
void producer()
{
    mutex=wait(mutex);
    full=signal(full);
    empty=wait(empty);
    x++;
    printf("\nProducer produces the item %d",x);
    mutex=signal(mutex);
}
```

```
void consumer()
{
```

```

mutex=wait(mutex);
full=wait(full);
empty=signal(empty);
printf("\nConsumer consumes item %d",x);
X--;
mutex=signal(mutex);
}

```

## OUTPUT:

The screenshot displays a C++ IDE with two windows. The left window shows the source code for a Dining Philosopher problem implementation, and the right window shows the program's output.

**Source Code (producer\_consumer.cpp):**

```

36 int wait(int s)
37 {
38     return(--s);
39 }
40
41 int signal(int s)
42 {
43     return(++s);
44 }
45
46 void producer()
47 {
48     mutex=wait(mutex);
49     full=signal(full);
50     empty=wait(empty);
51     x++;
52     printf("\nProducer produces the item %d",x);
53     mutex=signal(mutex);
54 }
55
56 void consumer()
57 {
58     mutex=wait(mutex);
59     full=wait(full);
60     empty=signal(empty);
61     printf("\nConsumer consumes item %d",x);
62     X--;
63     mutex=signal(mutex);
64 }
65

```

**Output:**

```

1.Producer
2.Consumer
Enter your choice1
Producer produces the item 1
Enter your choice1
Producer produces the item 2
Enter your choice1
Producer produces the item 3
Enter your choice1
Buffer is full
Enter your choice2
Consumer consumes item 3
Enter your choice2
Consumer consumes item 2
Enter your choice2
Consumer consumes item 1
Enter your choice2
Buffer is empty
Enter your choice_

```

The output shows the sequence of actions: the producer produces three items, and the consumer consumes them in reverse order (3, 2, 1). The program handles buffer full and empty conditions correctly.