# Producer Consumer Problem

## Introduction

This algorithm uses ADA to solve the producer/consumer (or bounded-buffer) problem.

## Algorithm

This process (task, to ADA) manages the buffer.

```
 1  task boundedbuffer is
 2          entry deposit(data: in item);
 3          entry extract(data: out item);
 4  end ;
 5  task body boundedbuffer is
 6          buffer: array [0..n-1] of item;
 7          count: integer range 0..n := 0;
 8          in, out: integer range 0..n-1 := 0;
 9          begin
10          loop
11                  select
12                          when count < n =>
13                                  accept deposit(data: in item) do
14                                          buffer[in] := data;
15                                  end ;
16                                  in := (in + 1) mod n;
17                                  count := count + 1;
18                          or when count > 0 =>
19                                  accept extract(data: out item) do
20                                          data := buffer[out];
21                                  end ;
22                                  out := (out + 1) mod n;
23                                  count := count - 1;
24                  end select ;
25          end loop ;
26  end .
```

The producer deposits an item into the buffer with

```
27          boundedbuffer.deposit(nextp);
```

and the consumer extracts an item from the buffer with

```
28          boundedbuffer.extract(nextc);
```

## Comments

lines 1-4 — This indicates that the procedures `deposit` and `extract` may be called outside the task, and that `extract` will return something in its parameter list (the **`out`**).

lines 6-8 — As usual, `buffer` is the buffer, and `count` the number of items currently in the buffer; `in` and `out` are the indices indicating where deposits go or where extractions come from.

lines 13-17 — If there is room in the buffer (**`when`** `count < n`) this process will accept a request to deposit an item in it (**`accept`** `deposit` ...); it then updates its variables.

lines 18-23 — If there is an item in the buffer (**`when`** `count > 0`) this process will accept a request to extract an item from the buffer (**`accept`** `extract` ...); the item is returned via the parameter list. This procedure then updates its variables.

line 24 — If both of the above two **`when`** conditions are true, and both a producer and consumer has invoked a procedure named by an **`accept`** statement (called "an open accept statement"), the system will select one to be executed in some fair manner (such as first-come-first-serve). If only one of the conditions is true, and the procedure named in an **`accept`** statement in the body of the when statement is open, that one will be executed. If both of the **`when`** conditions are false, an error condition occurs (this usually terminates the process.)