CHALMERS

Example: circular buffer

Problem: Write a server task **`Circular_Buffer`** in Ada that handles a circular buffer with room for 8 data records of type **`Data`**.

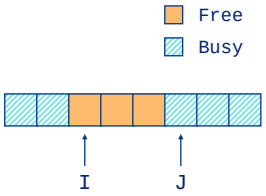
- The server task should have two entries, **`Put`** and **`Get`**.
- Producer tasks should be able to insert data records in the buffer via entry **`Put`**. If the buffer is full, a task that calls **`Put`** should be blocked.
- Consumer tasks should be able to remove data records from the buffer via entry **`Get`**. If the buffer is empty, a task that calls **`Get`** should be blocked.

CHALMERS

Example: circular buffer

```
task Circular_Buffer is
  entry Put(D : in Data);
  entry Get(D : out Data);
end Circular_Buffer;

task body Circular_Buffer is
  N : constant := 8;
  A : array (1..N) of Data;
  I,J : Integer range 1..N := 1;
  Count : Integer range 0..N := 0;
begin
  loop
    select
      when Count < N =>
      accept Put(D : in Data) do
        A(I) := D;    -- save data in buffer
      end Put;
      I := (I mod N) + 1;
      Count := Count + 1;
    or
      when Count > 0 =>
      accept Get(D : out Data) do
        D := A(J);    -- get data from buffer
      end Get;
      J := (J mod N) + 1;
      Count := Count - 1;
    end select;
  end loop;
end Circular_Buffer;
```



CHALMERS

Example: circular buffer

Problem: Write a protected object **`Circular_Buffer`** that handles a circular buffer with room for 8 data records of type **`Data`**.

- The protected object should have two entries, **`Put`** and **`Get`**.
- Producer tasks should be able to insert data records in the buffer via entry **`Put`**. If the buffer is full, a task that calls **`Put`** should be blocked.
- Consumer tasks should be able to remove data records from the buffer via entry **`Get`**. If the buffer is empty, a task that calls **`Get`** should be blocked.

CHALMERS

Example: circular buffer

```
type Buffer is array (Integer range <>) of Data;

protected type Circular_Buffer is
  entry Put(D : in Data);
  entry Get(D : out Data);
private
  N : constant := 8;
  A : Buffer(1..N);
  I,J : Integer range 1..N := 1;
  Count : Integer range 0..N := 0;
end Circular_Buffer;

protected body Circular_Buffer is
  entry Put(D : in Data) when Count < N is
  begin
    A(I) := D;
    I := (I mod N) + 1;
    Count := Count + 1;
  end Put;

  entry Get(D : out Data) when Count > 0 is
  begin
    D := A(J);
    J := (J mod N) + 1;
    Count := Count - 1;
  end Get;
end Circular_Buffer;
```