

7. Монитор Хоара

7.1. Общее описание

```
monitor <имя_монитора>;  
    <секция_описания>  
    <секция_инициализации>  
end <имя_монитора>.
```

**<секция_описания> ::= <описание типов,
констант, переменных, процедур,
функций>**

<секция_инициализации> ::= **begin
<инициализация переменных>**

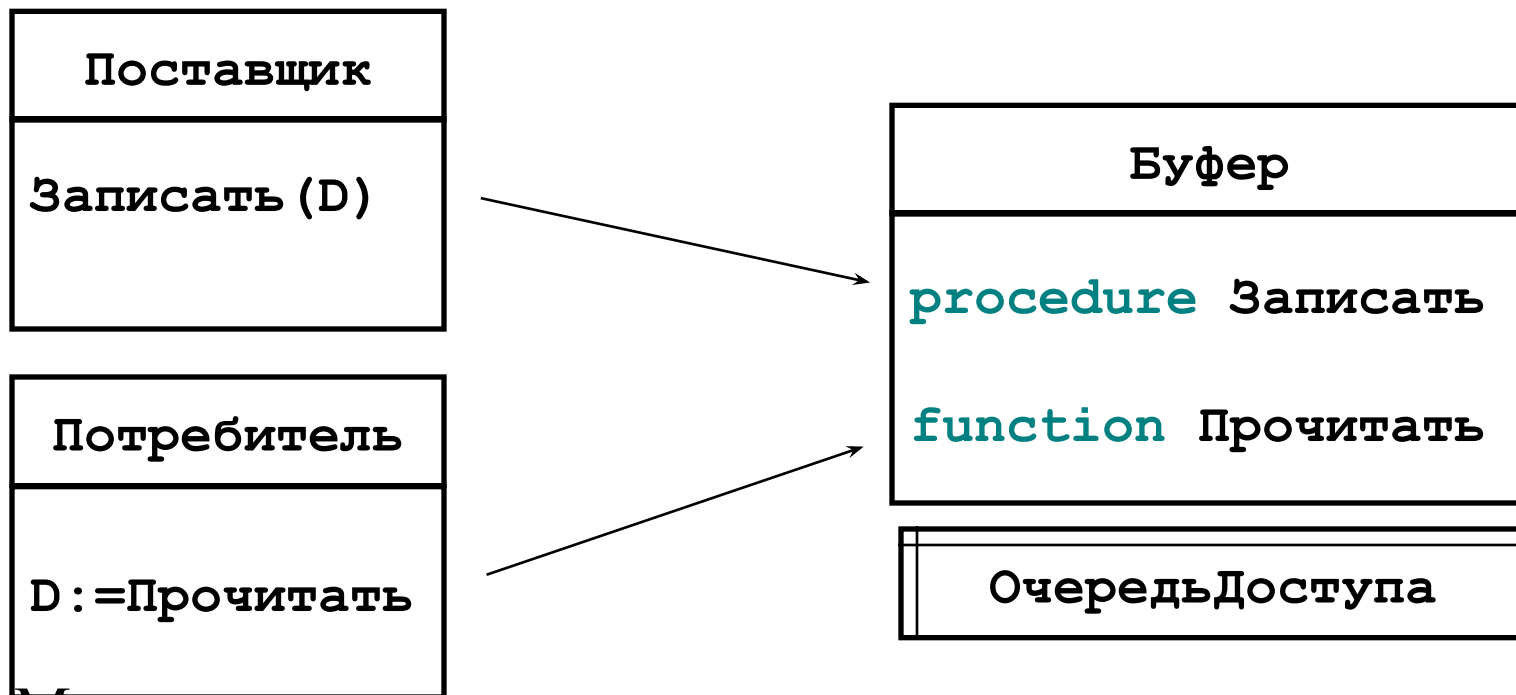
7.2. Пример

```
monitor Буфер;  
  constant ДлинаБуфера = 100;  
  var СамБуфер: array[1..ДлинаБуфера] of Данное;  
  var СчетчикЗаписей: integer;  
  procedure Записать(d: Данное);  
    begin  
      ...  
    end;  
  function Прочитать(): Данное;  
    begin  
      ...  
    end;  
begin  
  СчетчикЗаписей := 0;  
end Буфер.
```

7. Монитор

7.3. Процедура доступа

- Обеспечивается взаимное исключение доступа к ресурсам, описанным в мониторе (**Буфер.Записать (D)** и **Буфер.Прочитать ()** – не могут выполняться одновременно)
- Управление **ОчередьДоступа** (например, FIFO) обеспечивает очередность доступа процессов к монитору



7. Монитор

7.4. Процедура доступа

Поставщик:

```
loop
  D := Производство ();
  Буфер.Записать (D);
endloop.
```

Потребитель:

```
loop
  D := Буфер.Прочитать ();
  Обработка (D);
endloop.
```

```
monitor Буфер;
  . . .
  procedure Записать (d: Данное);
  begin
    . . .
  end;
  function Прочитать (): Данное;
  begin
    . . .
  end;
begin
  СчетчикЗаписей := 0;
end Буфер.
```

7. Монитор

7.4. Сигналы

Тип данных: `type Сигнал`

Операции над сигналами:

`Сигнал S = new Сигнал ();`

`wait(S) :`

поставить активный процесс в очередь, связанную с сигналом S

`send(S) :`

первый процесс из очереди S ставится в очередь готовых;

`check(S) : Integer :`

возвращает кол-во процессов, ждущих в очереди S

7. Монитор

7.5. Использование сигналов в мониторе

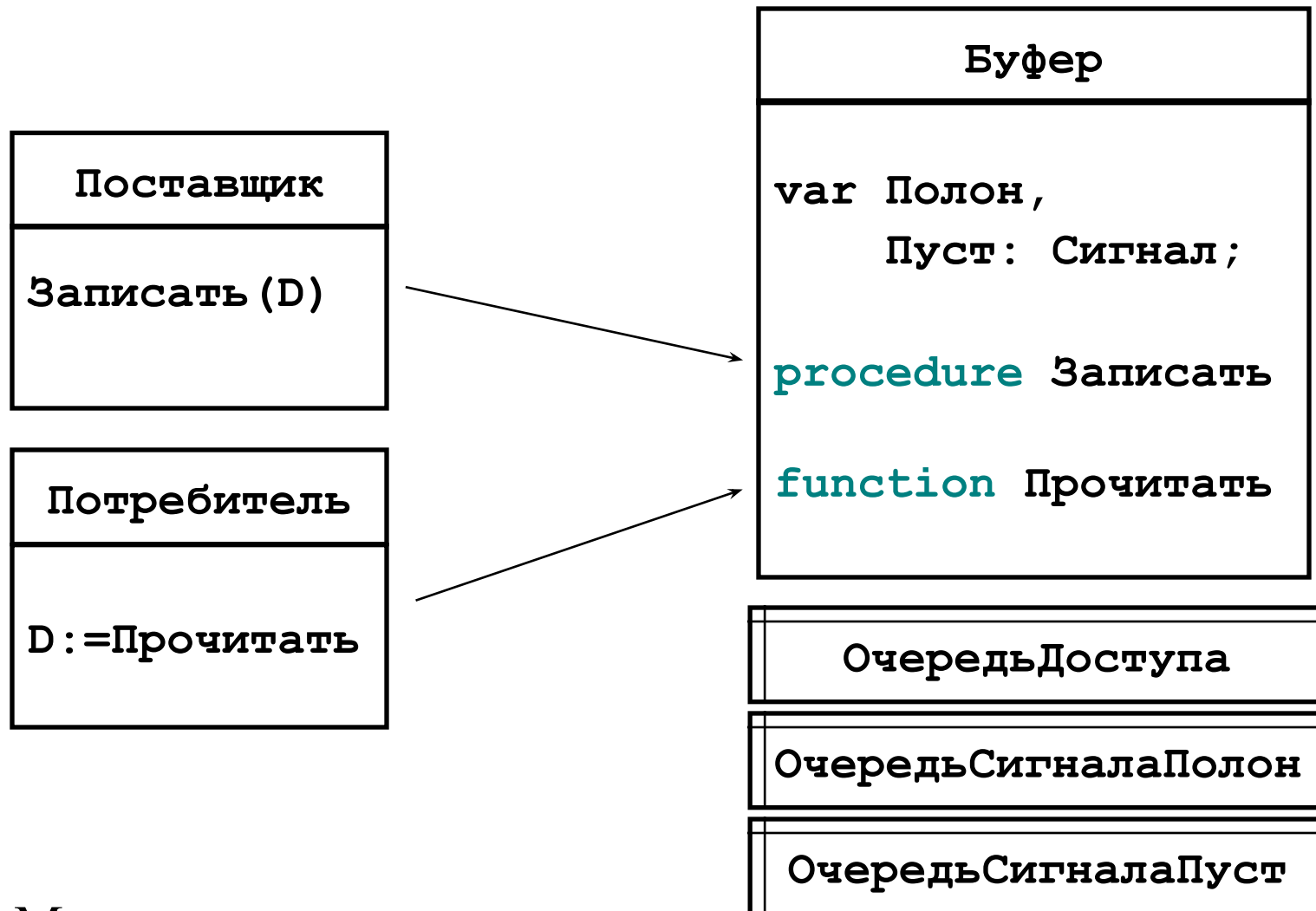
```
var Полон, Пуст: Сигнал;  
  
procedure Записать (d: Данное) ;  
begin  
    if (СчетчикЗаписей => ДлинаБуфера) then  
        wait (Полон)  
    endif  
    ЗаписатьДанноеВБуфер (d) ;  
    СчетчикЗаписей := СчетчикЗаписей + 1 ;  
    send (Пуст)
```

end;
7. Монитор

7.5. Использование сигналов в мониторе

```
function Прочитать () : Данное
begin
    if (СчетчикЗаписей = 0) then
        wait (Пуст)
    endif
    Прочитать := ЧтениеЗаписиИзБуфера ();
    СчетчикЗаписей := СчетчикЗаписей - 1;
    send (Полон)
end;
```

7.6. Очереди, связанные с монитором



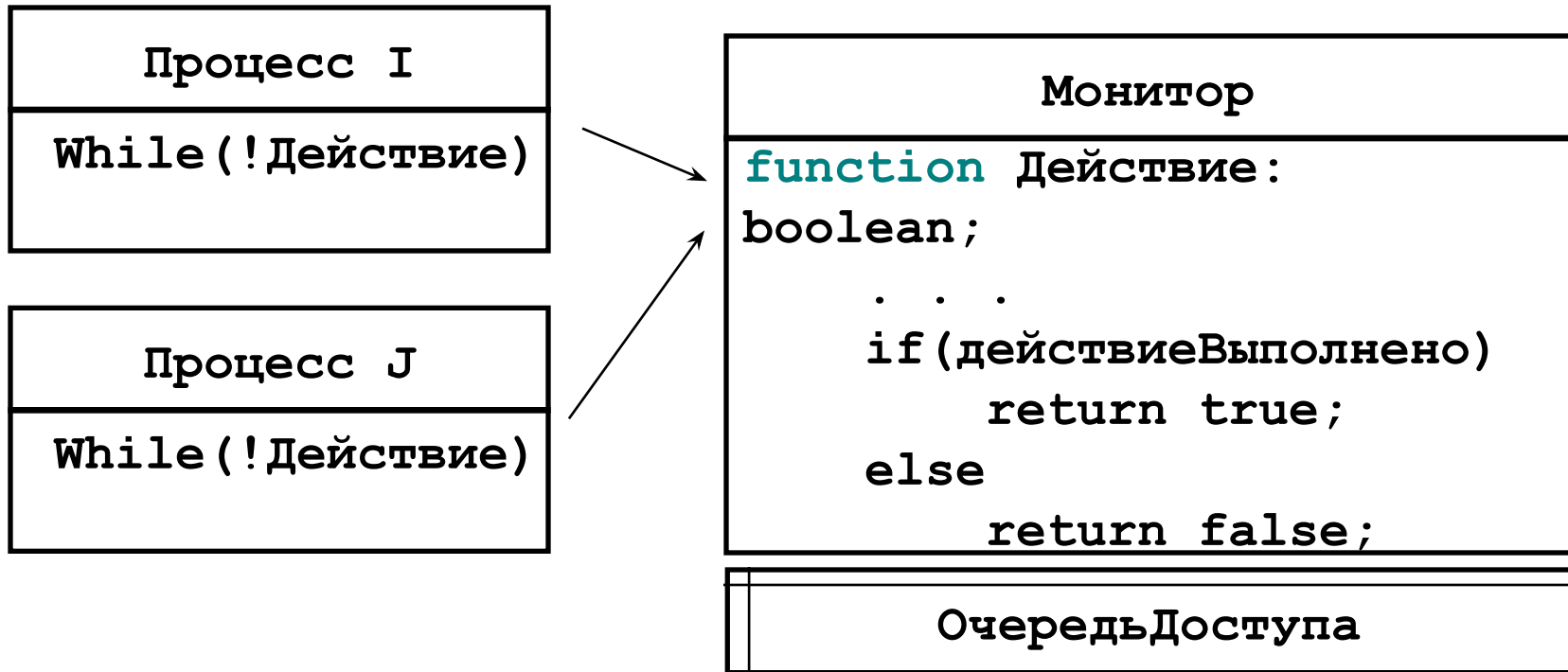
7. Монитор

7.6.1. Почему сигналы?

Допустим, сигналов нет. Тогда просматриваются только два способа ожидания возможности доступа к монитору

1. Процесс «сам анализирует» возможность доступа и «стучится» в монитор до тех пор, пока туда не попадет

7.6.1.1. Почему сигналы?



Плохо – приложение берет на себя функции платформы по распределению ресурса между процессами; приложения пишут разные люди, у которых разные интересы

7. Монитор

7.6.1.2. Почему сигналы?

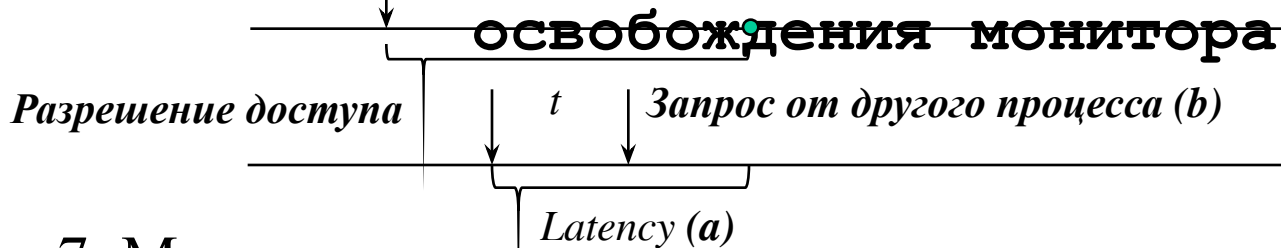
2. Процесс «сам анализирует» возможность доступа и «засыпает» на некоторое время (платформа предоставляет возможность «заснуть» на время t - `sleep(t)`):

```
while(!Действие) {  
    sleep(t);  
}
```

Еще хуже

a - Увеличивается latency

b - Можно «потерять» момент



7. Монитор

7.7. Пример 1 - задача «Читатели-Писатели»



- М Читателей и N Писателей получают доступ к Информационному Фонду
- Реализовать механизм, позволяющий обеспечить следующее условие:

в каждый момент времени могут работать не более одного Писателя или не более М Читателей

7. Монитор

7.7.1. Схема реализации

Читатель :

```
loop
    ЧП.НачалоЧтения ();
    РаботаСФондомЧ ();
    ЧП.КонецЧтения ();
    РазноеЧ;
endloop.
```

Писатель :

```
loop
    ЧП.НачалоЗаписи ();
    РаботаСФондомП ();
    ЧП.КонецЗаписи ();
    РазноеП;
endloop.
```

```
monitor ЧП;
var МожноЧитать,
    МожноПисать: Сигнал;
    КтоТоПишет: boolean;
    Читатели: 0..M;

procedure НачалоЧтения;
procedure КонецЧтения;
procedure НачалоЗаписи;
procedure КонецЗаписи;

begin
    КтоТоПишет:= false;
    Читатели:= 0;
end ЧП.
```

7.7.2. Начало чтения

```
procedure НачалоЧтения ();  
begin  
    if (КтоТоПишет) or (check (МожноПисать) > 0) then  
        wait (МожноЧитать)  
    endif  
    Читатели := Читатели + 1;  
    send (МожноЧитать)  
end;
```

7.7.3. Конец чтения

```
procedure КонецЧтения ();  
begin  
    Читатели := Читатели - 1;  
    if (Читатели = 0) then  
        send (МожноПисать)  
    endif  
end;
```

7.7.4. Начало записи

```
procedure НачалоЗаписи ();  
begin  
    if (Читатели > 0) or (КтоТоПишет) then  
        wait (МожноПисать)  
    endif;  
    КтоТоПишет := true;  
end;
```


7.7.5. Конец записи

```
procedure КонецЗаписи ();  
begin  
    КтоТоПишет := false;  
    if (check (МожноЧитать) > 0) then  
        send (МожноЧитать)  
    else  
        send (МожноПисать)  
    endif;  
end;
```

7.8. Пример 2 - реализация механизма семафоров через монитор

```
monitor Семафор;  
var Счетчик: 0..1;  
    S: Сигнал;  
  
procedure P;  
begin  
    if(Счетчик = 0) then  
        wait(S)  
    endif;  
    Счетчик = 0;  
end;
```

```
procedure V;  
begin  
    Счетчик := 1;  
    send(S);  
end;  
  
begin  
    Счетчик := 1;  
end Семафор.
```

7.8. Пример 2 - реализация механизма семафоров через монитор

Задача Ri:

```
loop
    . . .
    Семафор.P ();
    КритическаяСекция_i
    Семафор.V ();
    . . .
endloop.
```

Parbegin

```
R1 ;
R2 ;
. . .
Rn
```

Parend.

7. Монитор