## Лекция № 5

### Ресурсы, конфликты на ресурсах

Процессы Zi в системе Q развиваются параллельно. Это значит, что они изменяют значения параметров системы в течение одного и того же интервала времени. Достаточно типичны ситуации, когда по логике функционирования системы накладываются ограничения на изменение некоторых параметров несколькими процессами одновременно в течение заданного либо обусловленного интервала времени. Это возможно лишь для пересекающихся объектов. Ограничения развития процесса, накладываемые другими процессами, назовем конфликтными ситуациями, или конфликтами.

Общую область параметров для пересекающихся процессов Ok и Om назовем ресурсом. Таким образом, ресурс $R_{k,m}$ определяется, как

$$\mathbf{R_{k,m} = O_k \cap O_m}$$

Конфликт возможен только при наличии ресурса. Таким образом, необходимо добиться согласования процессов в этой области. В основе способов разрешения конфликтов лежит утверждение об обязательном разделении доступа процессов к ресурсу во времени. Рассмотрим следующие способы разрешения конфликтных ситуаций.

А. Явное разделение по времени (синхронизация). При этом способе разрешения конфликта разнесение во времени производится явным указанием интервалов времени, определенных для каждого процесса. На рисунке 4.9 показан пример выделения таких интервалов для случая конфликта трех процессов.

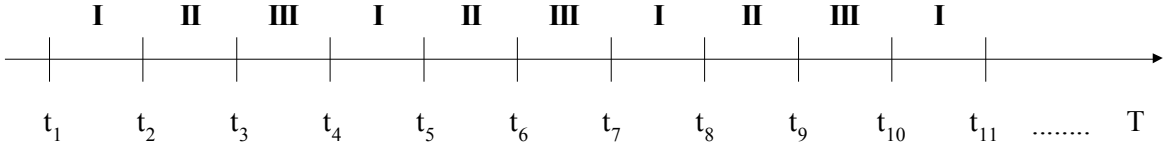
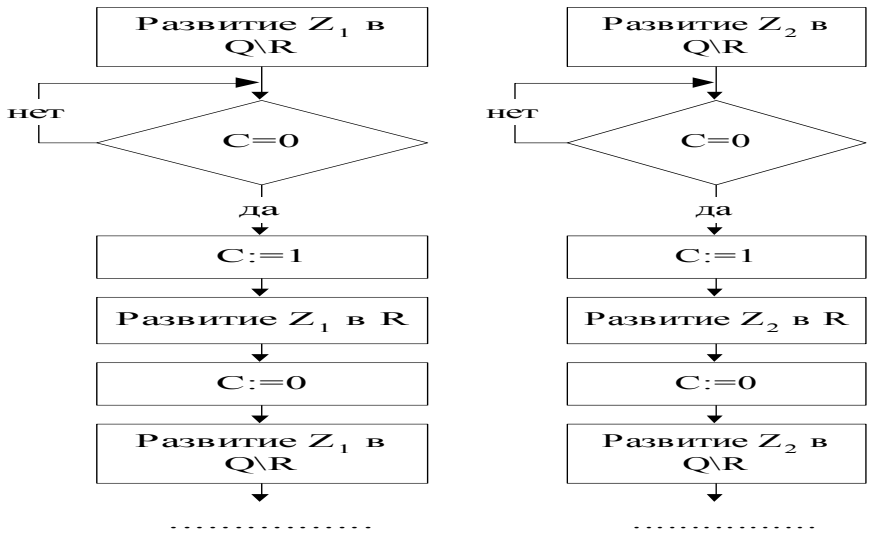



Рисунок 4.9. Разделение по времени

Б. Логический способ. В частности – использование семафоров. Если условие захвата ресурса не ограничивает время использования этого ресурса захватившим его процессом, то в этом случае удобно использовать семафоры. Семафор относится к логическим способам разрешения конфликтов. Семафор есть простая логическая переменная, однозначно соответствующая ресурсу. Значение семафора ‘0’ означает, что ресурс может быть захвачен процессом, значение семафора ‘1’ блокирует захват ресурса. На рисунке 4.10 показан пример использования семафора C при захвате ресурса R двумя процессами Z1 и Z2. Следует заметить, что в качества семафора может выступать любая логическая функция.

Рисунок 4.10. Использование семафоров




В. Алгоритмический способ. Наиболее общий способ управления процессами при захвате ресурсов состоит в использовании блоков - контроллеров. Использование этих блоков для разрешения конфликтов приведено ниже при определении понятия К-блока.

## ПОСП: Объекты языка, описания

**Основные положения**

В языке ПОСП разрешается использование букв и символов любых алфавитов, в выражениях могут использоваться знаки и символы любых операций из области математики, лингвистики и пр.

*Идентификатор* - последовательность букв, цифр и некоторых символов, начинающаяся с буквы. Набор допустимых символов определяется самим пользователем.

Типы идентификаторов:

а) *простой* - последовательность букв и цифр, начинающаяся с буквы;

б) *составной* - последовательность простых идентификаторов, соединенных символом подчеркивания;

в) *стандартный* - фиксированный простой или составной идентификатор. Стандартные идентификаторы могут быть пользовательскими и системными. Перечень пользовательских стандартных идентификаторов определяется самим пользователем.

В арифметических выражениях в качестве стандартных пользовательских идентификаторов часто используются имена функций таких, как SIN, COS, EXP, SQRT и т.д. Перечень таких идентификаторов может быть задан пользователем для каждой конкретной программы.

В качестве системных стандартных идентификаторов в языке ПОСП определены следующие:

ВРЕМЯ - переменная типа скаляр, ее значение - текущее время в модели;

RAND - стандартная функция (оператор). Вычисляет очередное значение псевдослучайной переменной **RAND** в [0,1] с равномерным законом распределения.

ИНИЦИАТОР - переменная типа ссылки, принимающая значение ссылки на локальную среду текущего процесса.

*Запись операторов языка сопровождается служебными словами, которые пишутся русскими буквами и выделяются* ***жирным шрифтом*** *(в рукописных текстах – подчеркиваются).* Полный текст оператора завершается символом **; .**

Запись чисел соответствует общепринятым в математике правилам.

Любой оператор может быть помечен меткой. *Метка* - идентификатор; метка отделяется двоеточием от оператора.

*Список* - линейная последовательность элементов, разделенных запятой.

*Строка символов* - любая последовательность символов, кроме кавычек, помещенная в кавычки.

Объекты

Объекты: *простая переменная; переменная; блок; инициатор.*

Каждый объект имеет тип, имя и значение.

*Имя объекта* есть идентификатор.

*Значение объекта* есть его содержание:

для простой переменной и переменной - логические, арифметические, текстовые или адресные значения;

для инициатора - адрес локальной среды процесса;

для блока - описание параметров блока и его алгоритм.

*Типы объектов*

*Тип простой переменной*:

**скаляр; ссылка; метка.**

*Тип переменной*:

**скаляр; ссылка; метка; вектор;.**

*Тип блока*:

**агрегат; процессор; контроллер, параметры.**

*Тип инициатора***: ссылка.**

Описание типов

*Скаляр* - одно неделимое значение. Это - число, логическое значение либо строка символов.

Синтаксис:

<список имен простых переменных> **- скаляр(ы)**

*Ссылка* - простая переменная типа скаляра, значением которой является адрес объекта.

Синтаксис:

<список имен простых переменных> **- ссылка**(**и**)

*Метка* - простая переменная типа скаляр, значением которой является адрес помеченного ею оператора.

Синтаксис:

<список имен простых переменных> **- метка**(**и**)

*Вектор* – линейно - упорядоченная совокупность скаляров либо векторов (определение рекурсивно).

Синтаксис:

<список имен переменных> **- вектор (**<список элементов описания вектора>**)**

<элемент описания вектора> **::=** <левая граница> **-** <правая граница> **-** < тип>

<левая граница>,<правая граница> **::=** целое число

<тип> **::= скаляр** | **ссылка** | **метка** | **вектор** |

Если левая и правая границы совпадают, то указание правой границы может быть опущено.

**Примеры**

1. Структура данных, приведенная на рис.1., может быть описана как:

**вектор** (1-3 - **скаляр**, 4-5 - **ссылка**, 6-7 - **скаляр**)

| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|

| скаляры | ссылки | скаляры |
|---|---|---|

Рис.5.1

Структура данных – **вектор**

2. Описание структуры, представленной на рисунке 5.2, выглядит как

**вектор (** 1**-вектор (** 1**-скаляр ,** 2**-вектор (**1-2**-ссылки ) ,** 3**-ссылка),** 2-3**-скаляры,** 4**-вектор (**1**-вектор (**1-3**-скаляры ),** 2**-ссылка ),** 5**-ссылка,** 6**-вектор (** 1-4**-скаляры ) )**

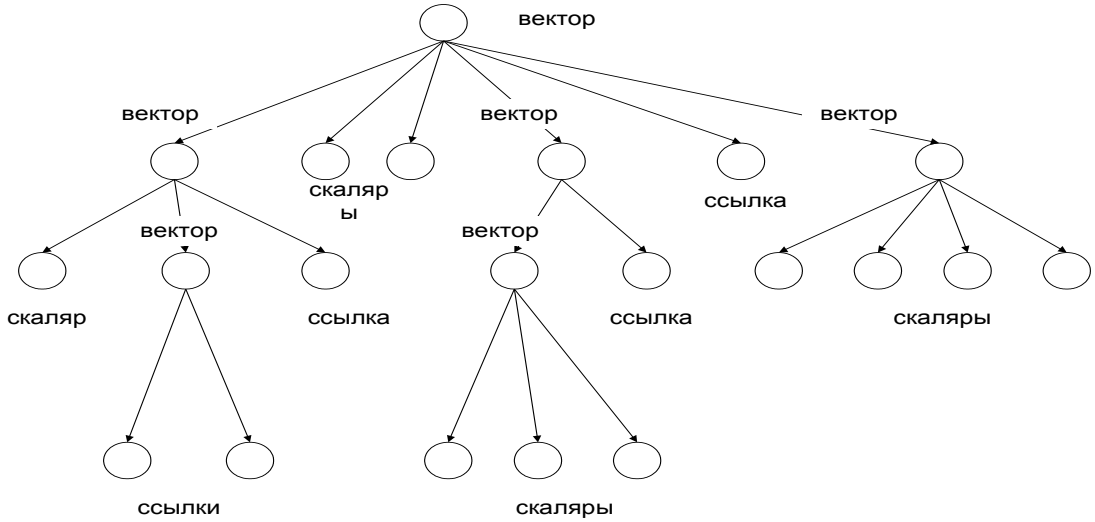



Рисунок 5.2. Пример структура данных - **вектор**

**Операторы ПОСП**

<программа> - последовательность операторов ПОСП.

**Операции над параметрами**

Над параметрами могут выполняться любые арифметические и логические операции.

*Арифметическое* (*логическое*) *выражение* есть последовательность арифметических (логических) операций над параметрами, имеющая целью получить некоторое конкретное числовое (логическое) значение.

Полученное значение присваивается переменной с помощью ***оператора присваивания****:*

<список переменных> **:=** <выражение>**;**

**Операции над инициатором**

*Навигационные операторы*

*Безусловный навигационный оператор*

Синтаксис:

**направить инициатор на** <метка> [ **блок** <имя блока>]**;**

Последняя часть может быть опущена, если эта операция происходит в одном и том же блоке.

*Условный навигационный оператор*

Синтаксис:

**если** <логическое выражение> **то направить инициатор на** <метка> [**иначе на** <метка> ]**;**

Последняя часть может быть опущена, если инициатор продолжает движение по программе.

*Векторная форма условного навигационного оператора*

Синтаксис:

**если**

B1 **направить инициатор на** <метка>

**:**

BN **направить инициатор на** <метка>

**[иначе направить на** <метка>**];**

где B1... BN - логические выражения.

*Оператор задержки инициатора*

*(оператор условия продвижения инициатора)*

Синтаксис:

**ждать** <логическое выражение>**;**

Оператор задерживает инициатор до выполнения логического условия. Условие может содержать и переменную **ВРЕМЯ**. Таким образом, этот оператор выполняет функции условного элементарного оператора, совмещая временной, логический и смешанный виды. Переменная ВРЕМЯ принимает в этом операторе абсолютное значение

*Векторная форма оператора задержки инициатора*

Синтаксис:

**ждать**

B1 **направить инициатор на** <метка>

**:**

BN **направить инициатор на** <метка>**;**

Здесь B1...BN - условные выражения.

Этот оператор совмещает функции двух последовательных операторов: оператора условия продвижения инициатора и векторного навигационного оператора. Его использование позволяет сократить запись в достаточно типичных конструкциях.

***Операторы над объектами***

*Оператор создания объекта*

Синтаксис:

**создать** <имя объекта> **типа** <тип>**;**

Оператор создает объект указанного типа и вводит его в программу.

*Оператор уничтожения объекта*

**уничтожить** <тип объекта> <имя объекта>**;**

Оператор уничтожает объект с заданным именем и выводит его из программы.

### Операции над ссылочными переменными

*Оператор присваивания значения ссылке.*

Синтаксис:

<имя ссылки> **:= ссылка на** [тип объекта] <имя объекта>

Ссылке присваивается адрес объекта.

*Оператор разыменования ссылки.*

Синтаксис:

<имя ссылки> → <тип переменной>[<место в переменной>]

Операция позволяет по ссылке определять значение переменной указанного типа. *Значение переменной* есть результат операции. Место в переменной необходимо указывать лишь для переменных типа **вектор**. *Если операция встречается в выражении, то вышеуказанная запись заключается в круглые скобки.*

**Примеры**

**1**. Вызов значения простой переменной типа скаляр, на которую ссылается ссылка S, имеет вид: S → **скаляр**

**2**. Вызов значения 5-го элемента вектора, на который ссылается ссылка S, имеет вид:

S → **вектор(5)**

**3**. Вызов значения 3-го элемента локальной среды текущего процесса имеет вид:

**ИНИЦИАТОР** → **вектор**(3)

Напомним, что значение инициатора есть ссылка на локальную среду процесса.

Комментарии могут вводиться в любом месте программы и отделяться от операторов двойным дефисом или двойным слешем ( -- , // )