Информатика

Лекция 4

Структурный подход к проектированию алгоритмов и программ

Нисходящая разработка

- это проектирование сверху-вниз (top-down), декомпозиция и пошаговая разработка алгоритмов

  (декомпозиция – представление сложного объекта в виде совокупности простых)

  - решение частных задач приводит к решению общей задачи
  - выбранная последовательность индивидуальных действий разумна
  - выбранная декомпозиция позволяет получить инструкции, в каком-либо смысле более близкие к языку, на котором в конечном счете будет сформулирована программа

Восходящая разработка

- это проектирование снизу-вверх (down-top), разработка алгоритмов путем объединения инструкций в элементарные процедуры («кластеры действий»)
- элементарные процедуры затем используются на следующем, более высоком уровне иерархии процедур

Структурное программирование

- *Разработка ясных, локально простых, удобочитаемых программ с древовидной структурой*
- *Программирование с использованием только базовых конструкций*

- **Основная теорема**
  Любая простая программа функционально эквивалентна структурированной программе, составленного из элементов базисного множества (следование, ветвление, цикл) с исполнением функций и предикатов исходной программы, а также присваиваний и тестов над дополнительным счетчиком

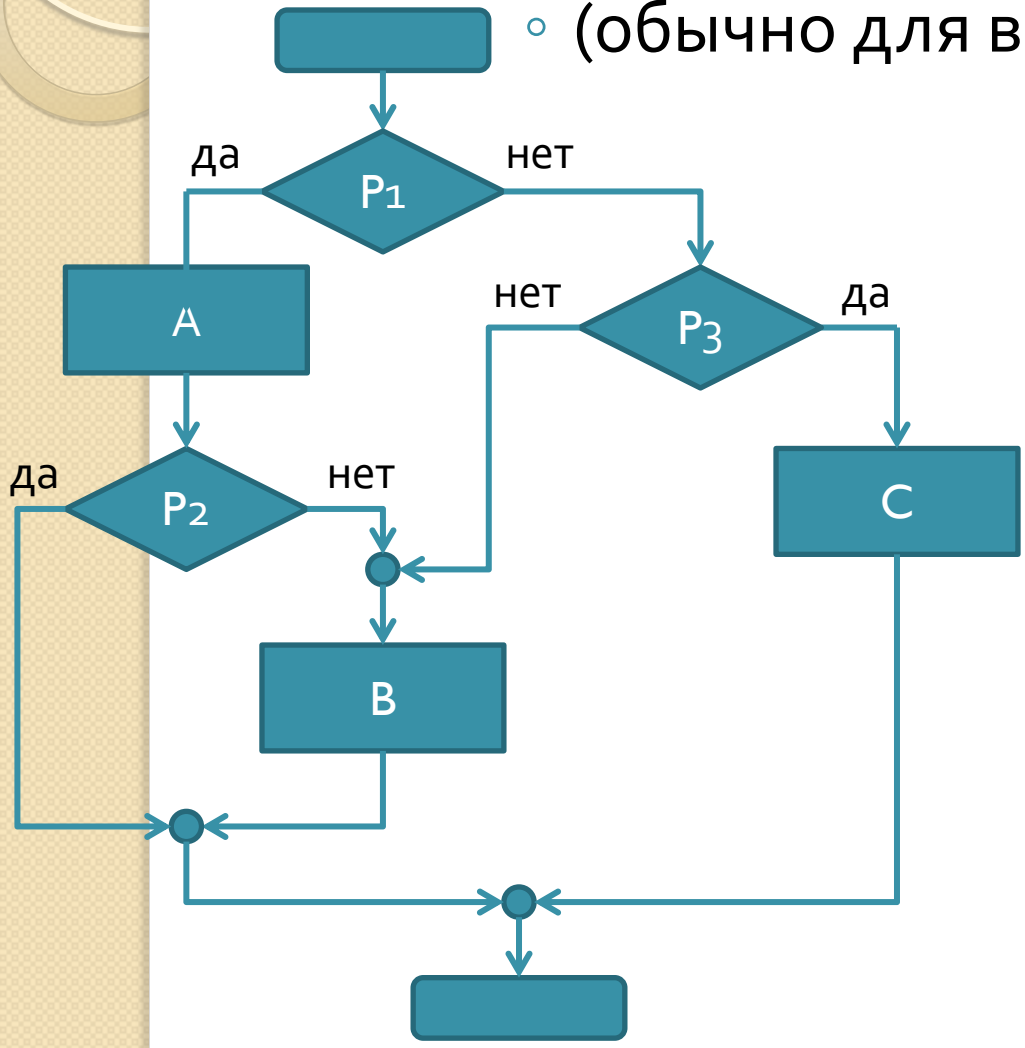
Обращение неструктурированных программ в структурированные

- дублирование процессов
- введение переменной состояния
- метод булевских функций

Эти три стратегии преобразования программ помогают овладеть методами проектирования и понимания алгоритмов и программ

Дублирование процессов (1)

- (обычно для ветвящихся процессов)

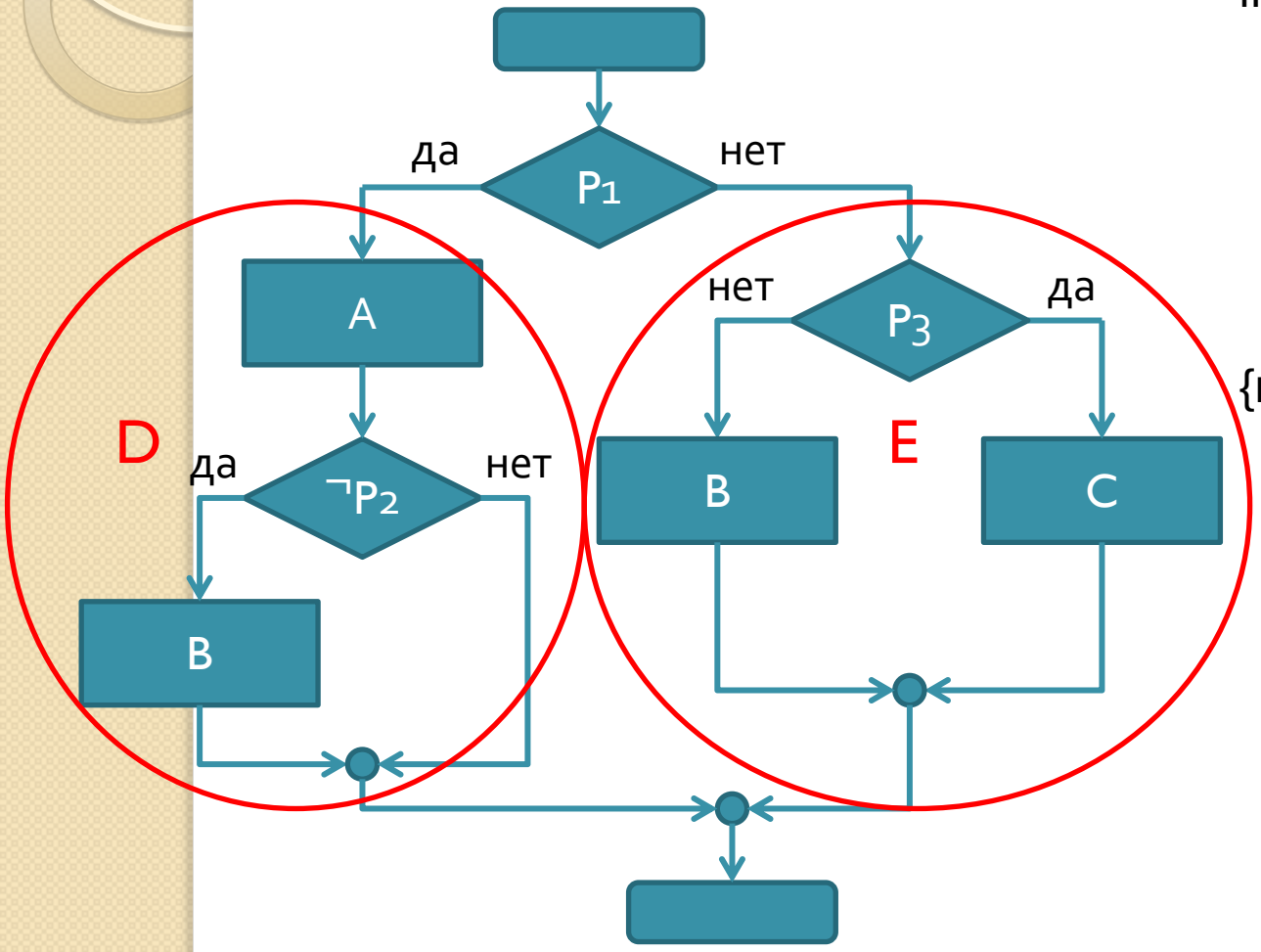


Тестируем возможные пути от входа к выходу:

P1 – да, P2 – да → A и выход
P1 – да, P2 – нет → A; B выход
P1 – нет, P3 – да → C → выход
P1 – нет, P3 – нет → B → выход

Дублируем процесс B, поскольку дублируются те процессы, в которые можно войти из нескольких мест

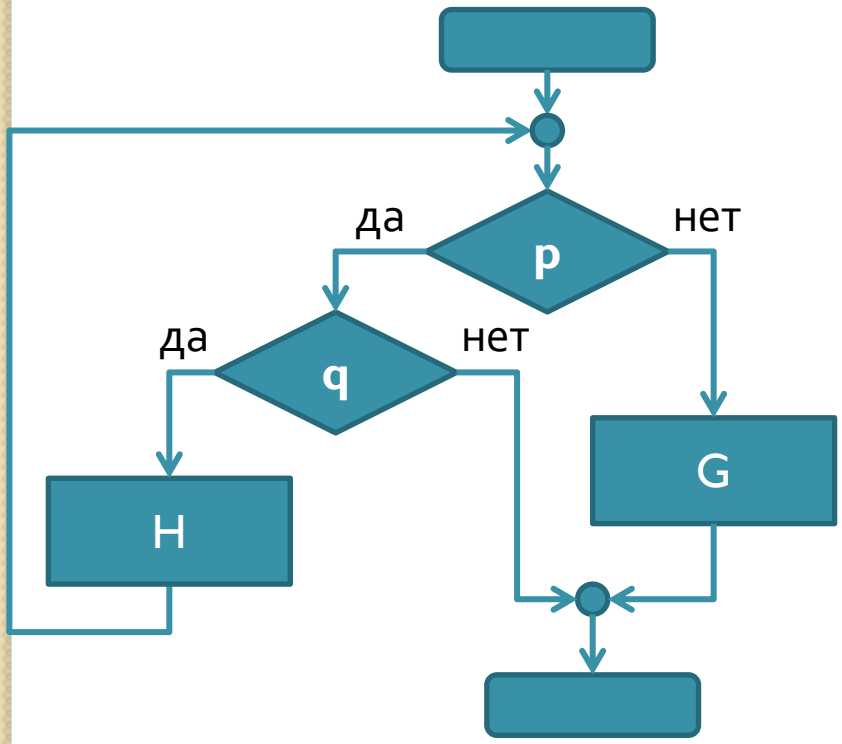
Дублирование процессов (2)




```
if P1 then
        begin
        A;
        if (not P2) then B
        end
        else
        if P3 then C else B;
{конец ветвления - if}
```

Введение переменной состояния

- Применим к любым программам и допускает автоматизацию преобразования

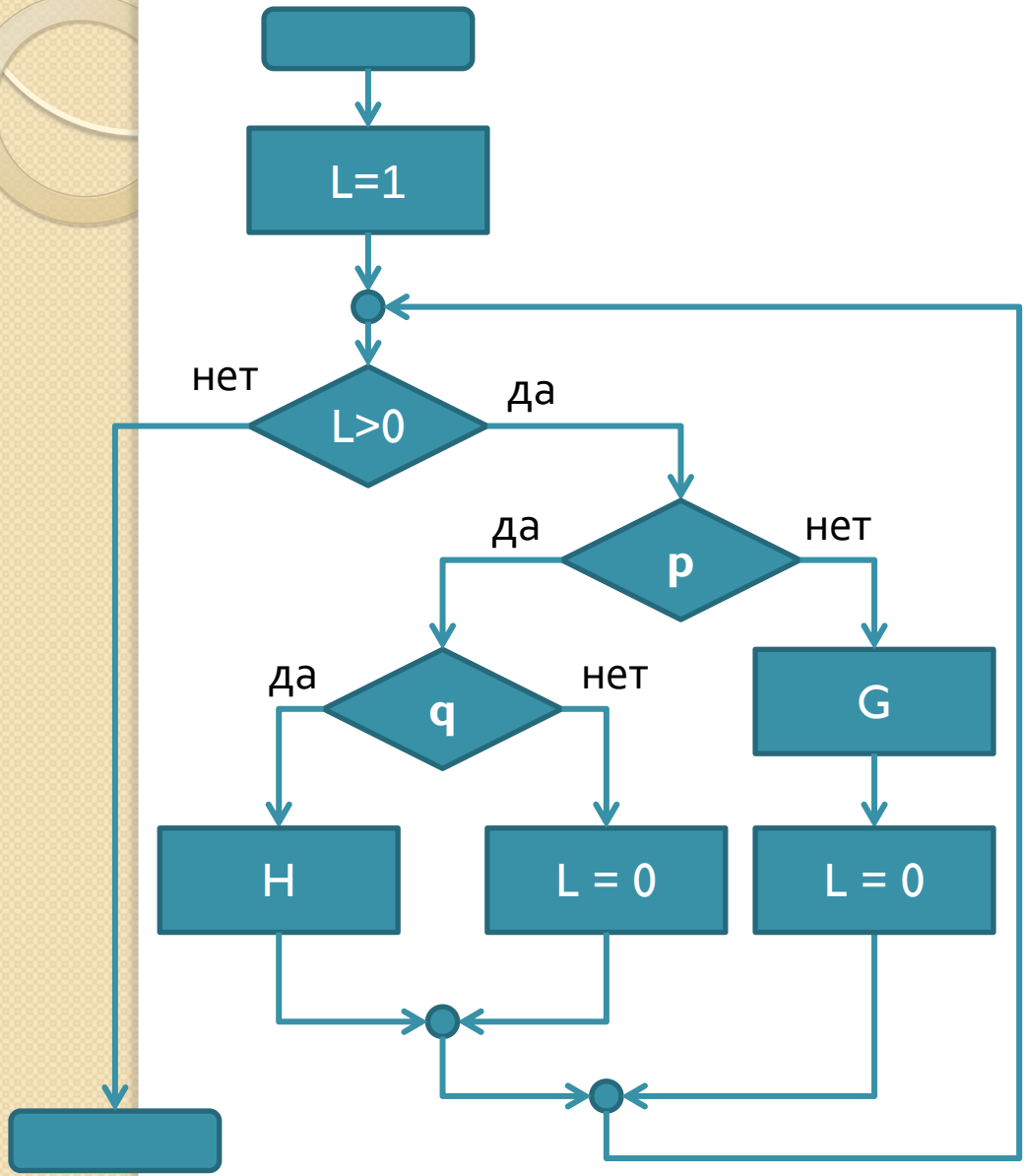


Тестируем возможные пути от входа к выходу:

p – нет, **q** – нет → G → выход
p – нет, **q** – да → G → выход
p – да, **q** – нет → выход
p – да, **q** – да → H → цикл (возврат к **p**)

Согласно теореме вводим счетчик L и выполняем тесты над счетчиком

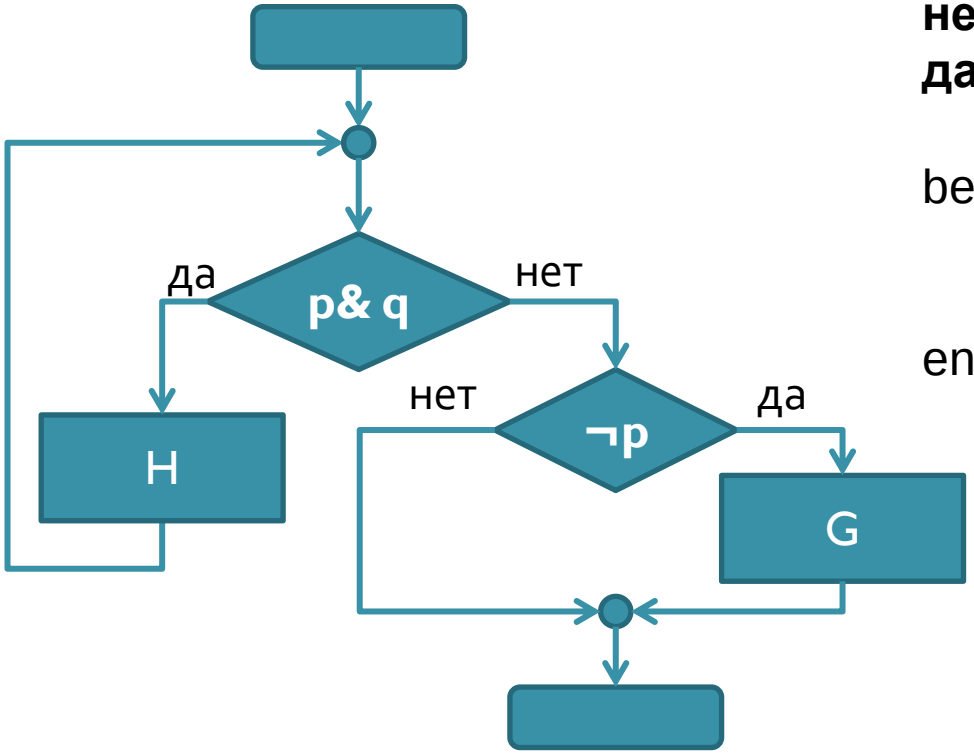
Введение переменной состояния (2)




```
begin
        L:=1;
        while L>0 do
                begin
                  if p then
                          if q then
                          H
                          else
                          L:=0
                  else begin
                  G; L:=0
                        end
                end;
```

Рассмотрим условие **p&q**
p – нет, **q** – любой → G, выход
p – да, **q** – да → цикл
с процессом H

Введение переменной состояния (3)



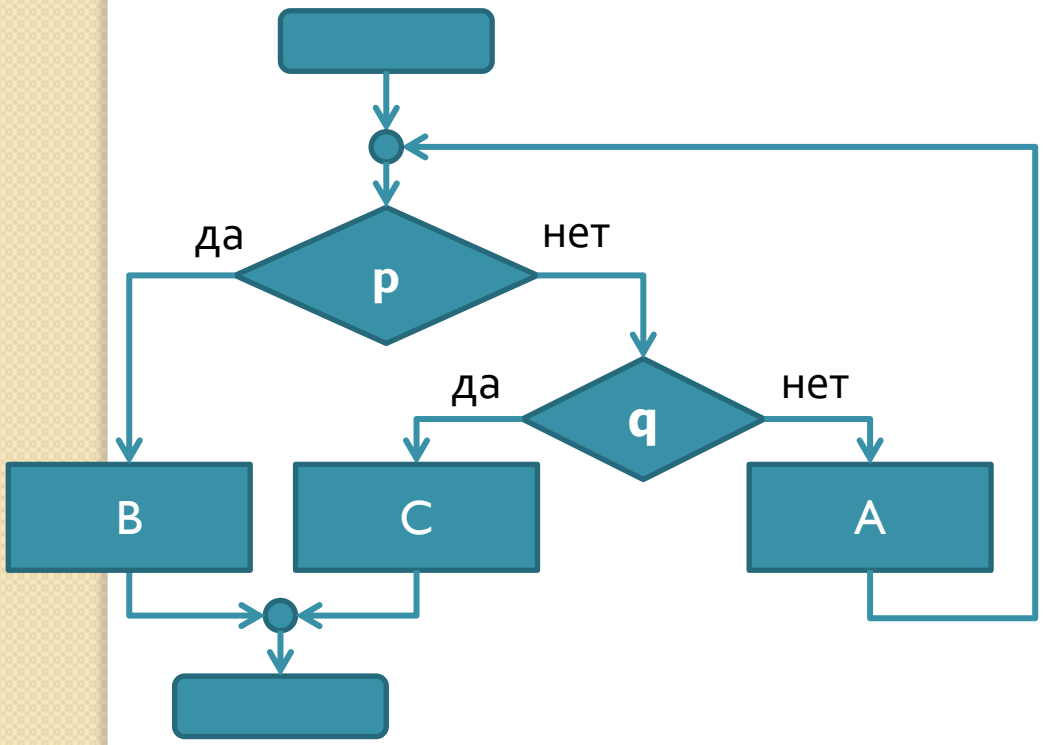
Рассмотрим условие **p & q**:

нет | **p** – нет, **q** – нет
нет | **p** – нет, **q** – да
нет | **p** – да, **q** – нет
да | **p** – да, **q** – да

```
begin
        while (p & q) do H;
        if (not p) then G;
end.
```

Метод булевской переменной (1)

- Может использоваться при преобразовании циклических процессов

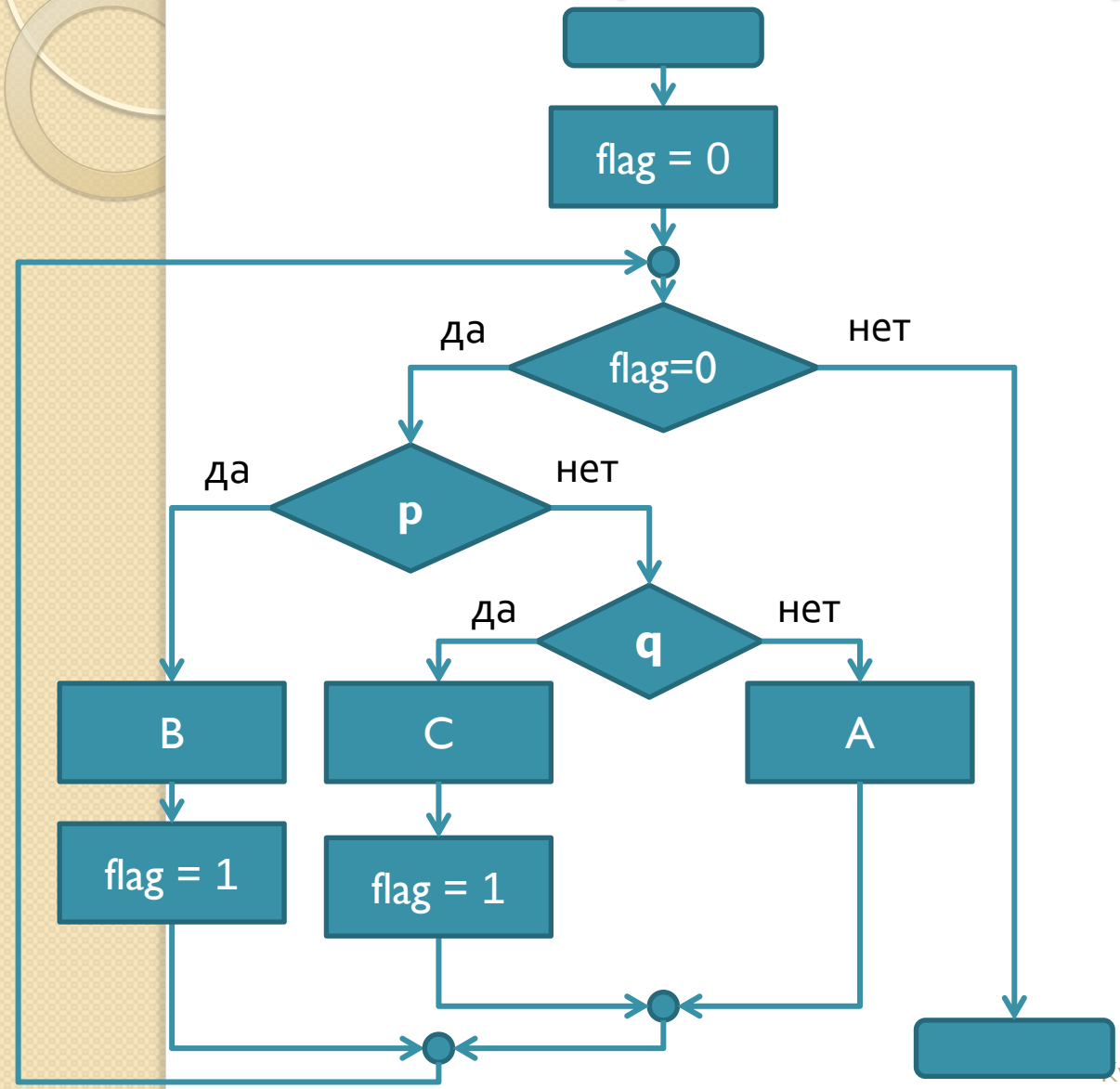


Тестируем возможные пути от входа к выходу:

p – да, q – да → B → выход
p – да, q – нет → B → выход
p – нет, q – да → C → выход
p – нет, q – нет → A и повтор на проверку p

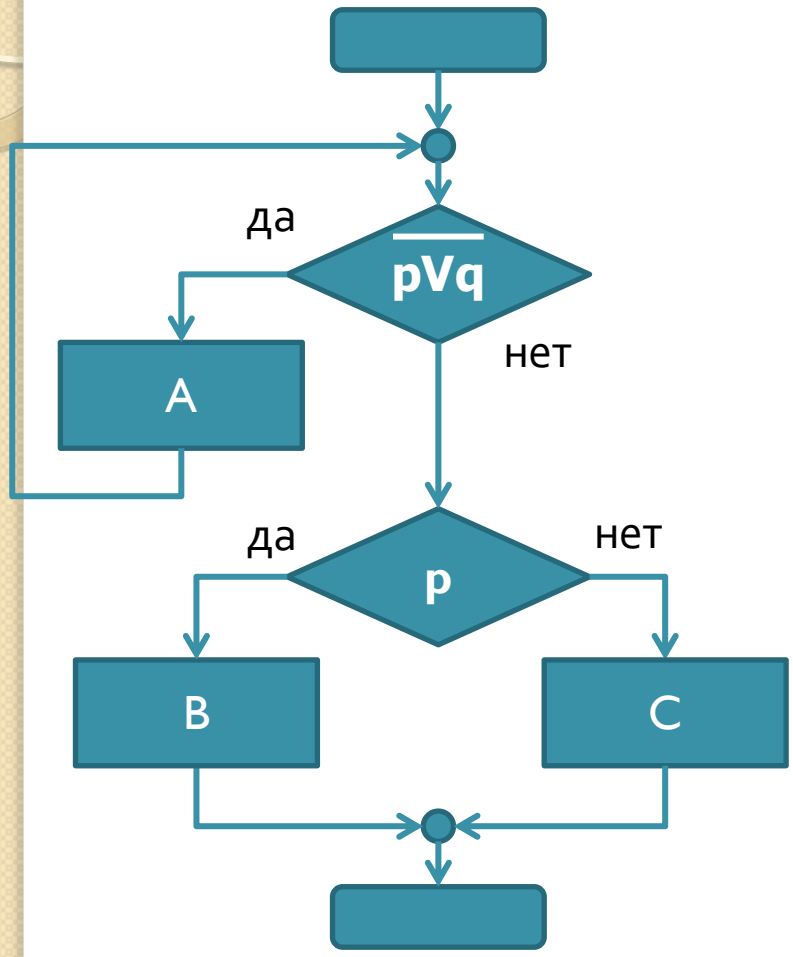
Досрочный выход из цикла по q – да

Метод булевской переменной (2)




```
begin
flag = 0;
while (flag=0) do
        begin
            if p then
               begin
               B;
               flag = 1
               end
            else
            if q then
               begin
               C;
               flag = 1
               end
            else
                A;
        end;
end.
```

Метод булевской переменной (3)



Рассмотрим условие **p V q**:

нет | **p** – нет, **q** – нет
да | **p** – нет, **q** – да
да | **p** – да, **q** – нет
да | **p** – да, **q** – да

```
begin
      while  not(p V q) do A;
      if (p) then B else C;
end.
```

Цикл лекций подготовлен в 2013/2014уч. году Кузнецовым Игорем Ростиславовичем, доцентом кафедры радиоэлектронных средств Санкт-Петербургского Государственного электротехнического университета «ЛЭТИ»

Прочитан в дисциплине
«Информатика»