Вложенный условный оператор

Вложенный условный оператор

Обсуждение:

Программирование умных рекомендаций

Поздравляем с профессиональным успехом!

Новость о программах для «Сладких историй» быстро распространилась в профессиональном сообществе.

Теперь к специалистам ProTeam обратился магазин здорового питания «Долголетие», желающий настроить рекомендации по категориям товаров **и** предпочтениям.

Кажется, эта задача похожа на предыдущую. Готовы попробовать?

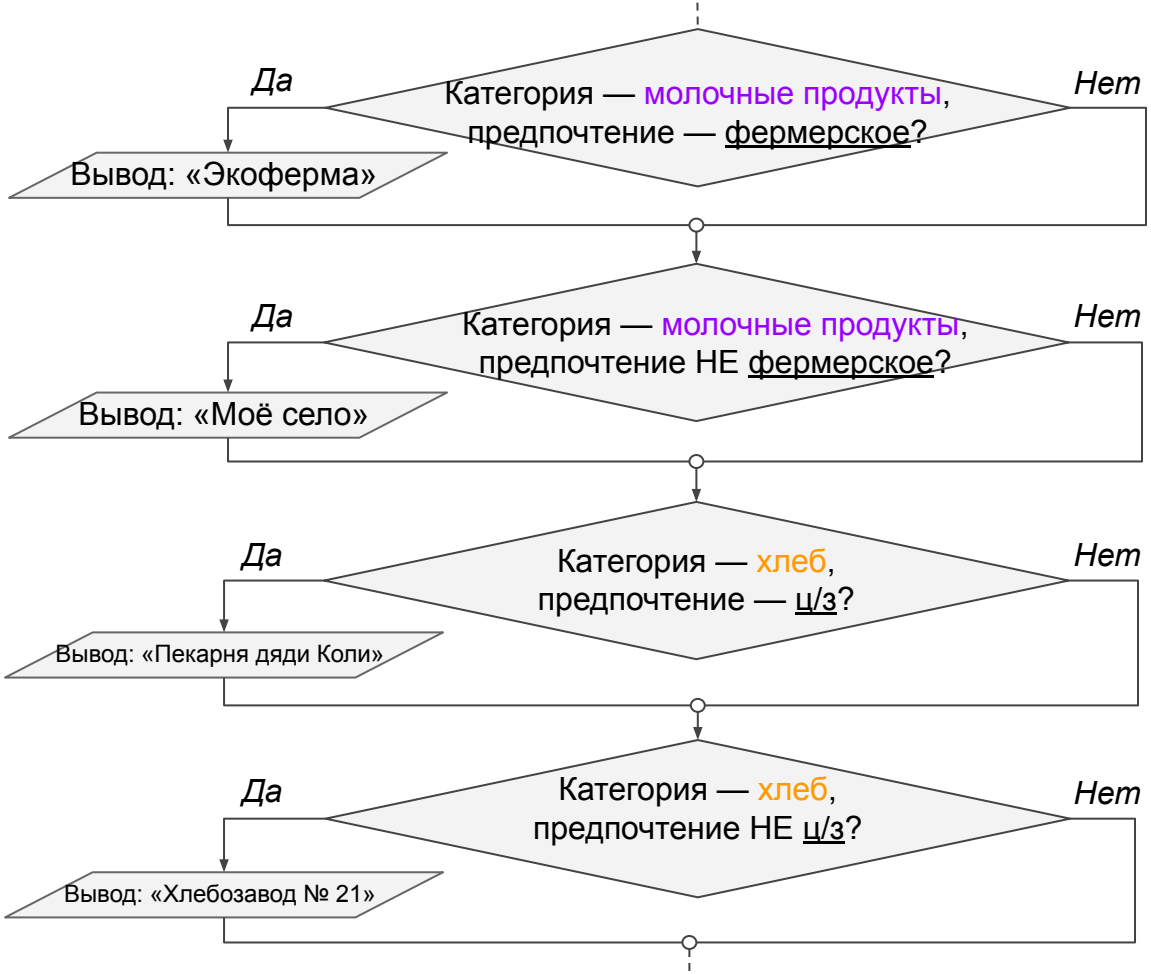
Многоуровневые умные рекомендации

Задача. Программа запрашивает у покупателя категорию товаров и предпочтения. После обработки ответов программа печатает рекомендованный товарный бренд. Для работы была передана таблица рекомендаций:

Категория	*Предпочтение*	*Рекомендация*
Молочные продукты	Фермерское	«ЭкоФерма»
Молочные продукты	Остальные предпочтения	«Моё село»
Хлеб	Цельнозерновой	«Пекарня дяди Коли»
Хлеб	Остальные предпочтения	«Хлебозавод № 21»

Как написать такую программу? Может, начнём с блок-схемы?

Возможна следующая схема работы программы:



Многоуровневые умные рекомендации

Задача. Программа запрашивает у покупателя категорию товаров и предпочтения. После обработки ответов программа печатает рекомендованный товарный бренд. Учитывая таблицу рекомендаций:

```
category = input('Категория товара:')
wish = input('Предпочтение:')
```

?

Как запрограммировать умные рекомендации?

Многоуровневые умные рекомендации

Задача. Программа запрашивает у покупателя категорию товаров и предпочтения. После обработки ответов программа печатает рекомендованный товарный бренд. Учитывая таблицу рекомендаций:

```
category = input('Категория товара:')
wish = input('Предпочтение:')
if category == 'молочные продукты' and wish == 'фермерское':
   print('Экоферма')
if category == 'молочные продукты' and wish != 'фермерское':
   print('Моё село')
if category == 'хлеб' and wish == 'цельнозерновой':
  print('Пекарня дяди Коли')
if category == 'хлеб' and wish != 'цельнозерновой':
  print('Хлебозавод №21')
```

Многоуровневые умные рекомендации

Задача. Программа запрашивает у покупателя категорию товаров и предпочтения. После обработки ответов программа печатает рекомендованный товарный бренд. Учитывая таблицу рекомендаций:

```
category = input('Категория товара:')
wish = input('Предпочтение:')
if category == 'молочные продукты' and wish == 'фермерское':
    print('Экоферма')
if category == 'молочные продукты' and wish != 'фермерское':
    print('Моё село')
if category == 'хлеб' and wish == 'цельнозерновой':
   print('Пекарня дяди Коли')
if category == 'хлеб' and wish != 'цельнозерновой':
   print('Хлебозавод №21')
```

Программа будет работать корректно.

Можно ли её **оптимизировать**?

Многоуровневые умные рекомендации

Задача. Программа запрашивает у покупателя категорию товаров и предпочтения. После обработки ответов программа печатает рекомендованный товарный бренд. Учитывая таблицу рекомендаций:

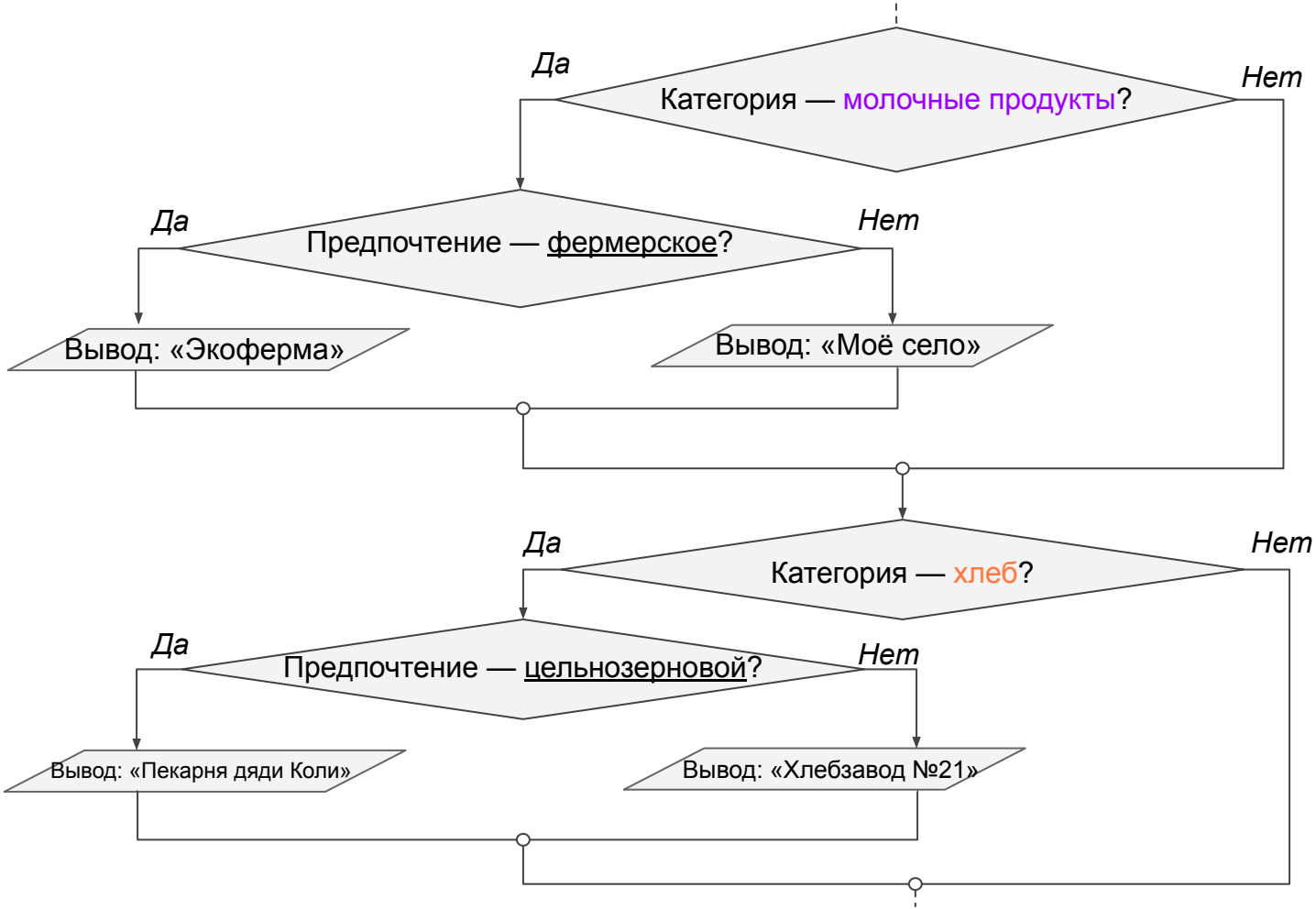
```
category = input('Категория товара:')
wish = input('Предпочтение:')
if category == 'молочные продукты' and wish == 'фермерское':
    print('Экоферма')
if category == 'молочные продукты' and wish != 'фермерское':
    print('Моё село')
if category == 'хлеб' and wish == 'цельнозерновое':
  print('Пекарня дяди Коли')
if category == 'хлеб' and wish != 'цельнозерновое':
  print('Хлебозавод №21')
```

Проверка категории повторяется.

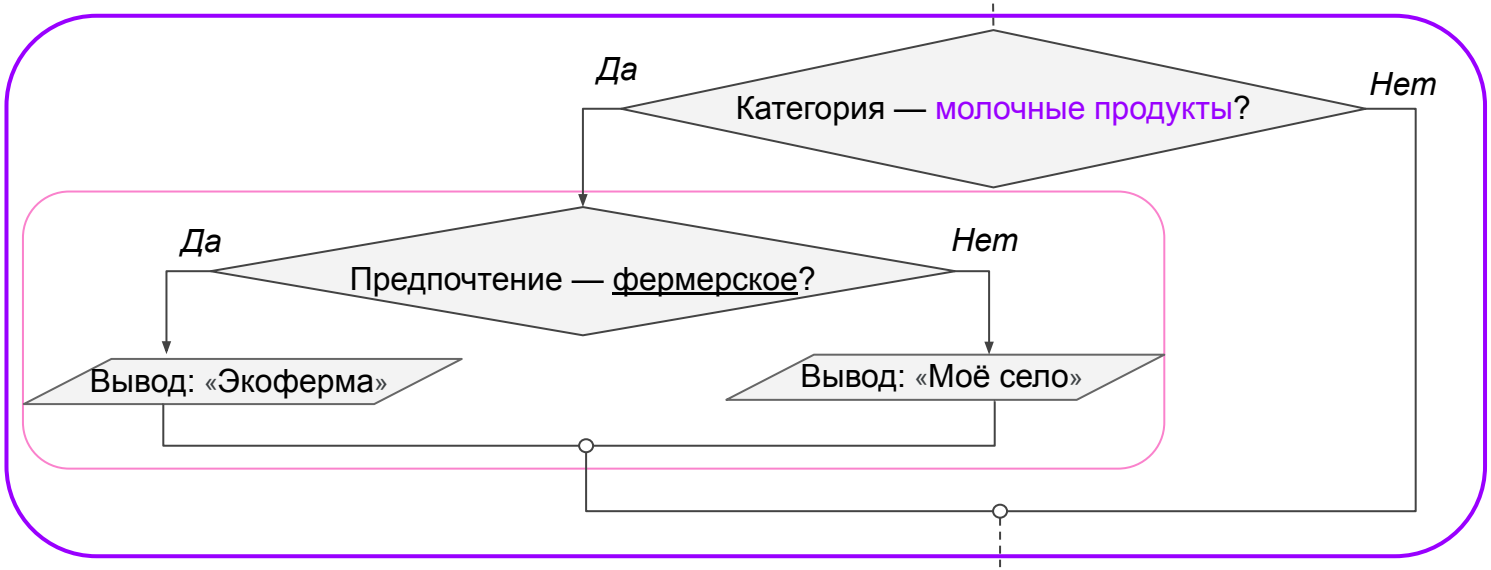
Можно ли **выполнить проверку один раз**, а не два?

Если да, как изменится блок-схема?

Возможна следующая схема работы программы:



Возможна следующая схема работы программы:



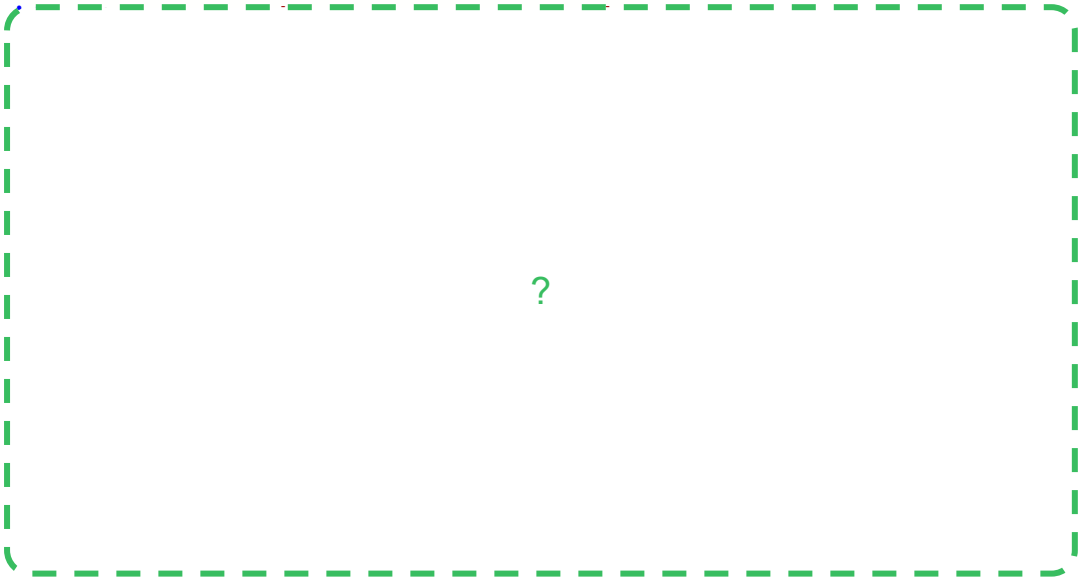
Напоминает вложенную конструкцию...

Как запрограммировать вложенность одного условного оператора в другой?

Многоуровневые умные рекомендации

Задача. Программа запрашивает у покупателя категорию товаров и предпочтения. После обработки ответов программа печатает рекомендованный товарный бренд. Учитывая таблицу рекомендаций:

```
category = input('Категория товара:')
wish = input('Предпочтение:')
```



Многоуровневые умные рекомендации

Задача. Программа запрашивает у покупателя категорию товаров и предпочтения. После обработки ответов программа печатает рекомендованный товарный бренд. Учитывая таблицу рекомендаций:

```
category = input('Категория товара:')
wish = input('Предпочтение:')
if category == 'молочные продукты':
    if wish == 'фермерское':
        print('Экоферма')
    else:
        print('Моё село')
if category == 'хлеб':
    if wish == 'цельнозерновой':
        print('Пекарня дяди Коли')
    else:
        print('Хлебзавод №21')
```

Оказывается, вложенность работает и для условного оператора!

Её использование позволяет избежать чрезмерно сложной проверки условий.

Цель рабочего дня —

настроить умные рекомендации *для магазина.*

Заказчик хочет, чтобы программа предлагала товары
и торговые марки в зависимости от предпочтений клиентов.

Сегодня вы:

- узнаете, что вложенный условный оператор —
  это средство программирования сложных
  условий;
- узнаете и запрограммируете условный оператор
  с несколькими ветвями;
- запрограммируете многоуровневые умные
  рекомендации.

Вложенный условный оператор

Повторение

Каким образом можно создать простое логическое выражение?

Какие значения оно может принимать (логический тип данных?

Простое логическое выражение

Логическое выражение принимает только значение `True` или `False`.

При составлении логических выражений могут использоваться операторы сравнения.

Логический тип					
>	<	==	!=	<=	>=
Больше	Меньше	Равно	Не равно	Меньше или равно	Больше или равно

Назовите значения выражений:

```
1 == 2
```

```
2 == 1 + 1
```

```
a == 5
```

```
15 == '15'
```

```
3.14 > 3
```

```
'Hello' != 'hello'
```

```
(3 + 2) * 0.1 == 0.5
```

Назовите значения выражений:

`1 == 2`	**False**	
`2 == 1 + 1`	**True**	
`a == 5`	**True**	Если значение переменной **a** равно 5, в других случаях False.
`15 == '15'`	**False**	Строка не равна числу.
`3.14 > 3`	**True**	
`'Hello' != 'hello'`	**True**	Две строки равны, только если все символы в них полностью совпадают.
`(3 + 2) * 0.1 == 0.5`	**True**	

С помощью каких операторов можно создать составное логическое выражение?

Составное логическое выражение

Составное логическое выражение можно создать **из простых**, связав их с помощью логических операторов:

Оператор	*Название*	*Используется когда нужно:*
and	Логическое И	потребовать выполнения двух простых условий одновременно
or	Логическое ИЛИ	потребовать выполнения хотя бы одного из двух простых условий

порядок выполнения

**Сначала выполняются части выражения, связанные логическим И, а потом — логическим ИЛИ.*

Назовите значения выражений:

```
'Да' == 'Да' and 3 > 2
```

```
1 > 2 and 3 > 2
```

```
1 > 2 or 3 > 2
```

```
ans == 'Да' and 2 == '2'
```

```
5 > 3 and 6 > 3
```

```
ans == 'Да' or ans != 'Да'
```

Назовите значения выражений:

`'Да' == 'Да' and 3 > 2`	**True**	
`1 > 2 and 3 > 2`	**False**	Первое выражение ложно (д. б. истинно первое ***и*** второе)
`1 > 2 or 3 > 2`	**True**	Второе выражение истинно (д. б. истинно первое ***или*** второе)
`ans == 'Да' and 2 == '2'`	**False**	Второе выражение ложно
`5 > 3 and 6 > 3`	**True**	
`ans == 'Да' or ans != 'Да'`	**True**	Действительно, значение переменной или равно некоторой величине, или нет.

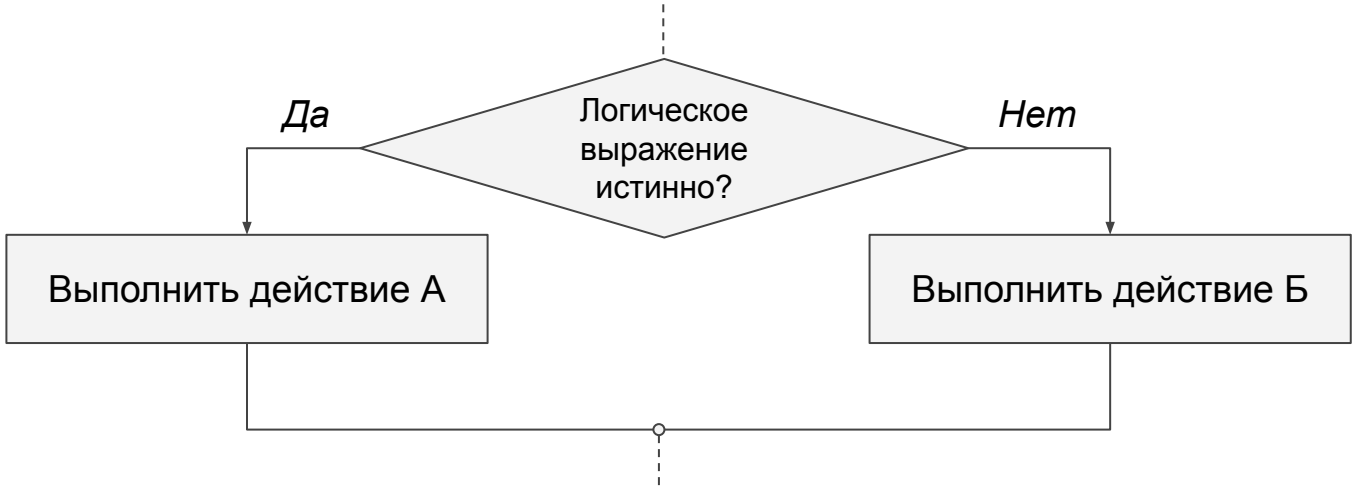
Что такое условный оператор?

Для чего он используется?

Условный оператор

— это команда, выполняющая или не выполняющая действие в зависимости от значения логического выражения.

Пример использования:
выполнение действия А если выражение истинно и действия Б — если ложно.



Как оформляется условный оператор?

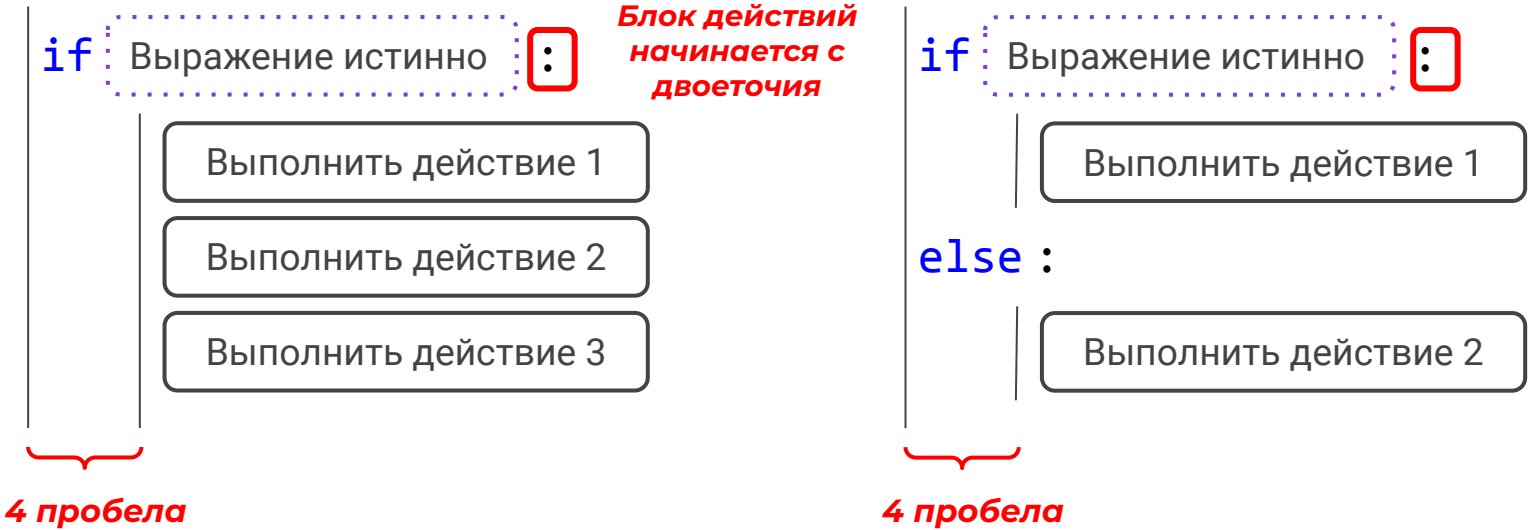
Какие служебные слова используются?

Условный оператор

Для программирования условного оператора используются команды:

`if` (*в англ. — «если»*)

`else` (*в англ. — «иначе»*)



Вложенный условный оператор

В чём особенность вложенного условного оператора?

Вы убедились, что вложенный условный оператор может быть заменён составным логическим выражением.

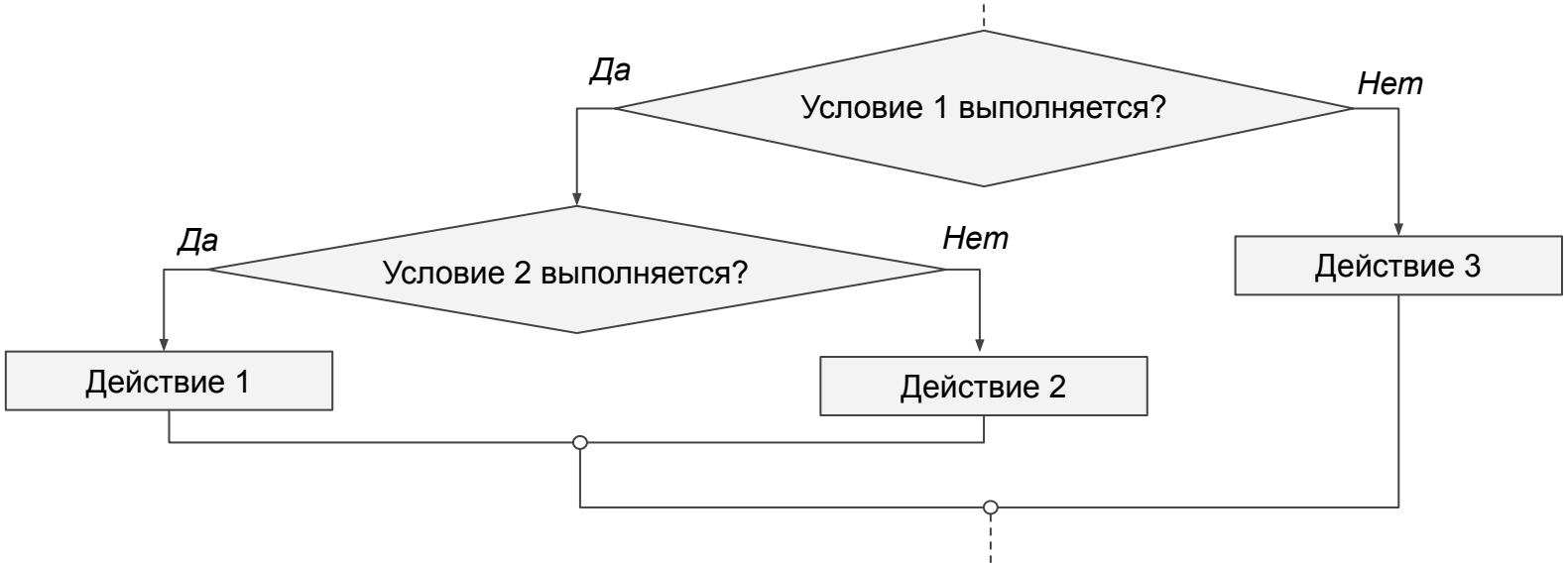
С другой стороны, использование вложенности:

- сокращает проверку условий;
- оптимизирует код программы;
- делает код более логичным и читаемым.

Вложенный условный оператор

Чтобы использовать вложенный условный оператор, нужно:

- продумать порядок проверки условий;
- уверенно знать оформление условного оператора.

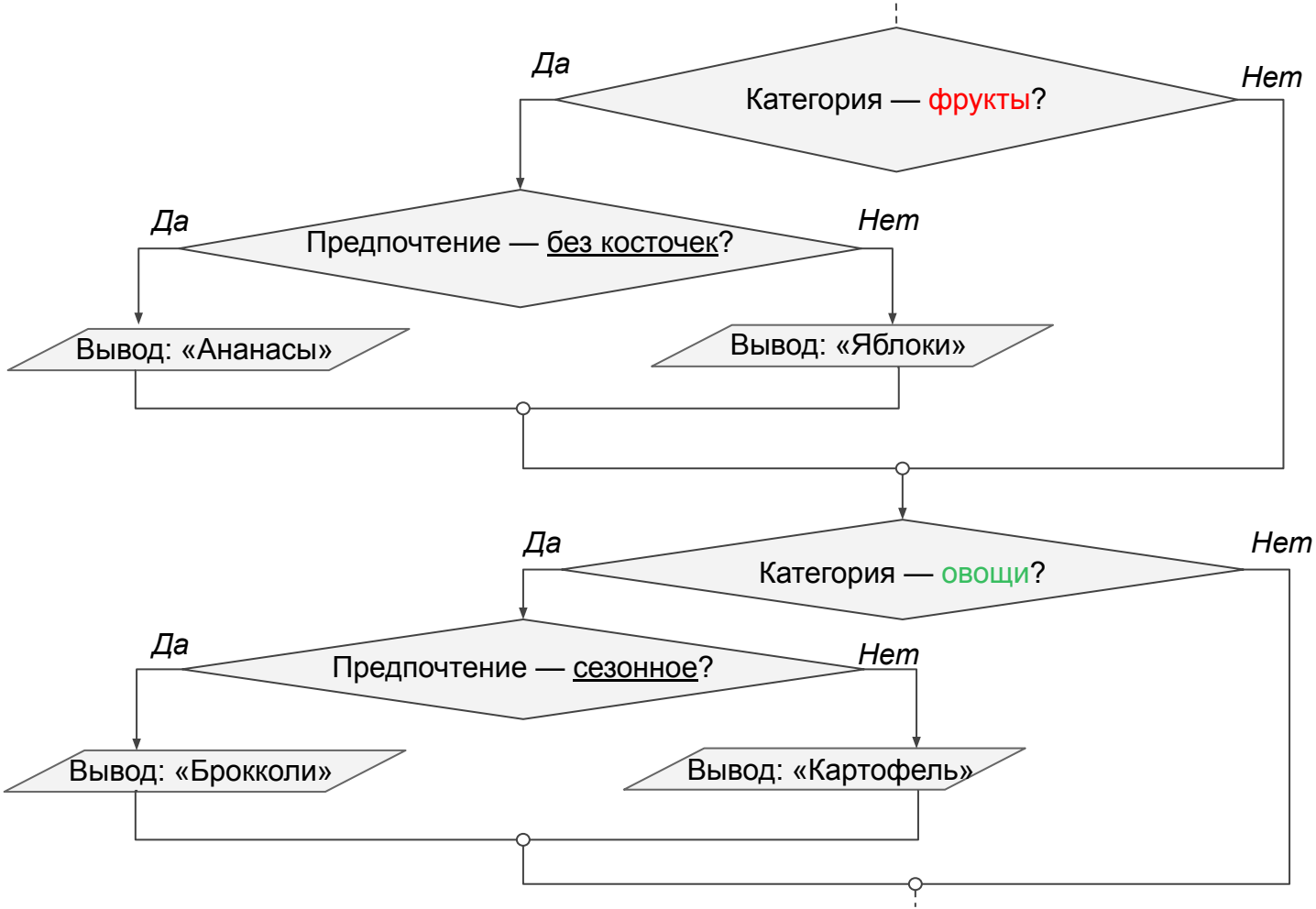


Порядок проверки условий

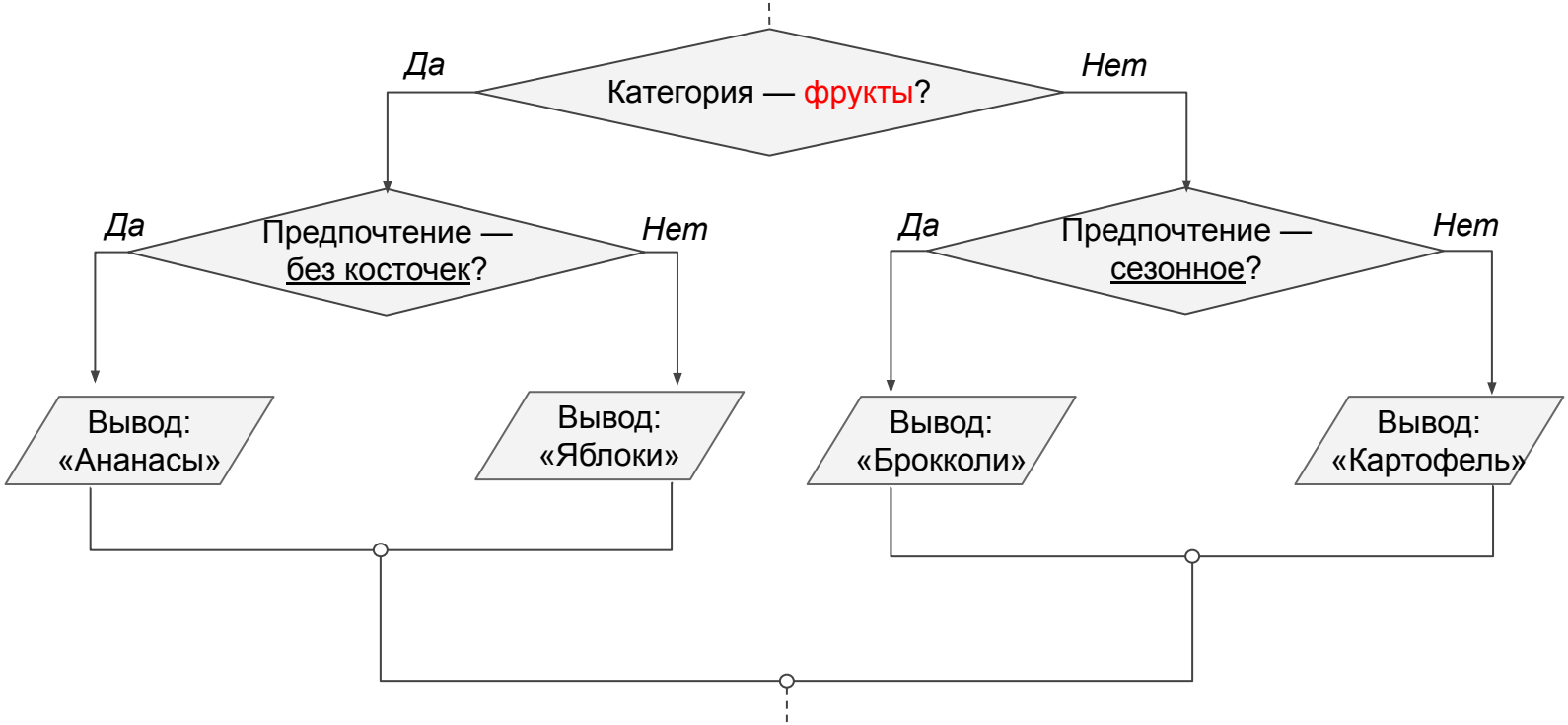
Задача. Составить алгоритм печати рекомендаций фруктов и овощей. Желающим фрукты без косточек печатать «Ананасы», остальным — «Яблоки». Желающим сезонные овощи печатать «Брокколи», остальным — «Картофель».

Какие условия можно выделить? Какое условие рационально проверить первым?

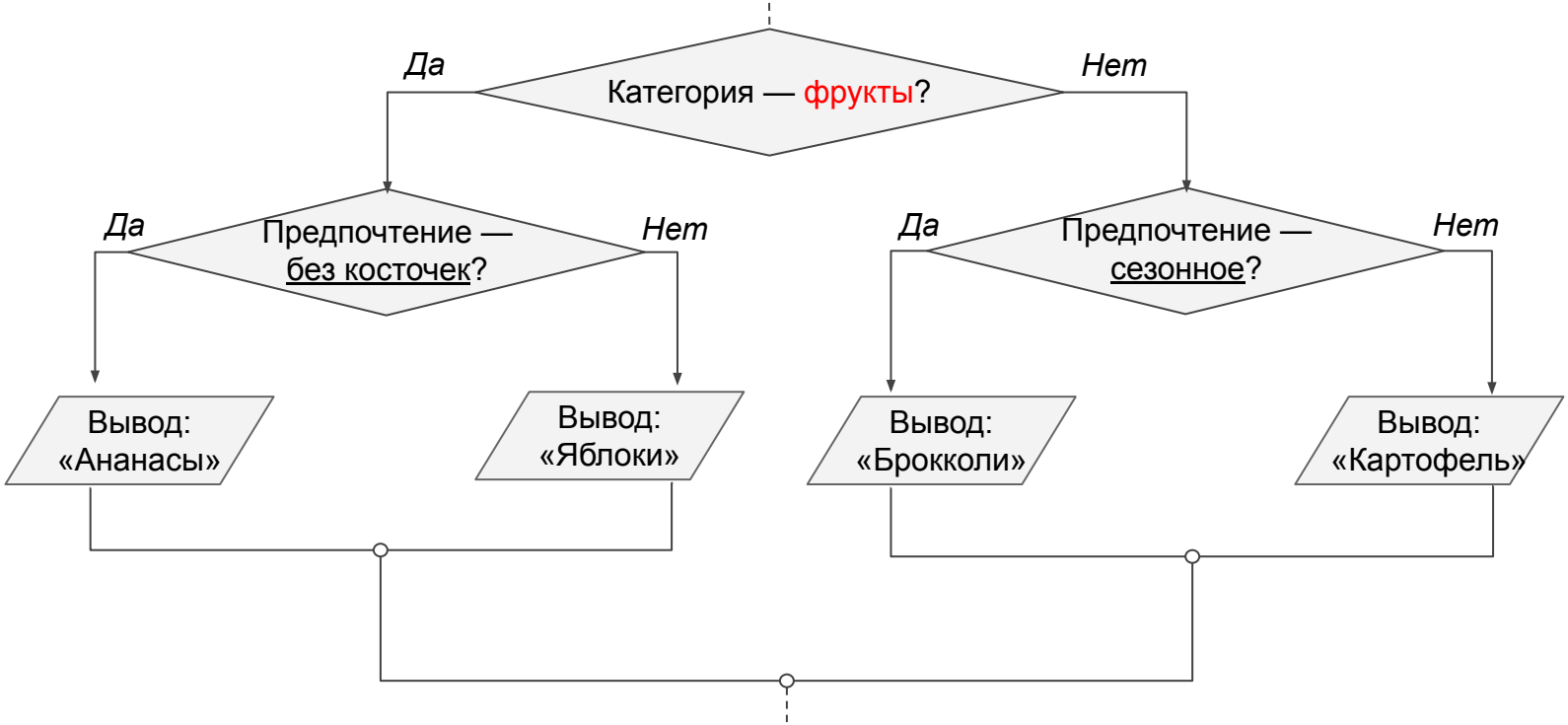
Возможный алгоритм работы программы:



Будет ли работать такой алгоритм?



Будет ли работать такой алгоритм?

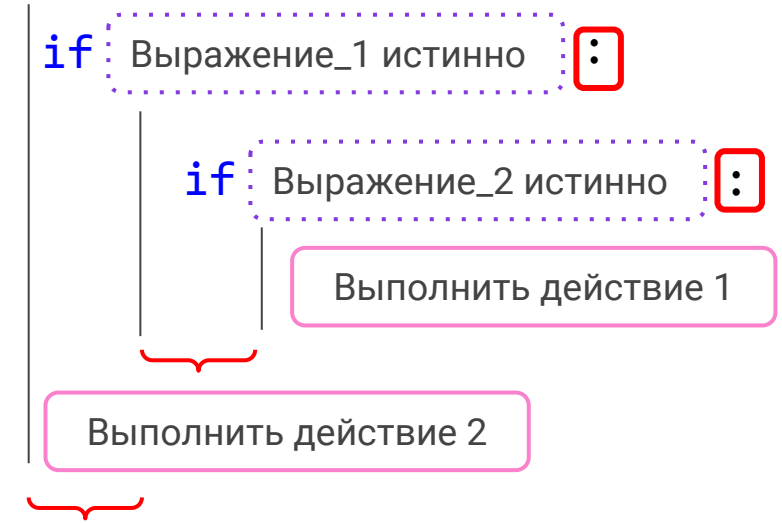


Да, но не всегда корректно.

Если покупатель напечатает категорию «Хлеб» и предпочтение «С семечками», то ему будет рекомендован картофель.

Оформление вложенности

Новых правил оформления нет. Нужно очень внимательно придерживаться уже известных.



Оформление вложенности

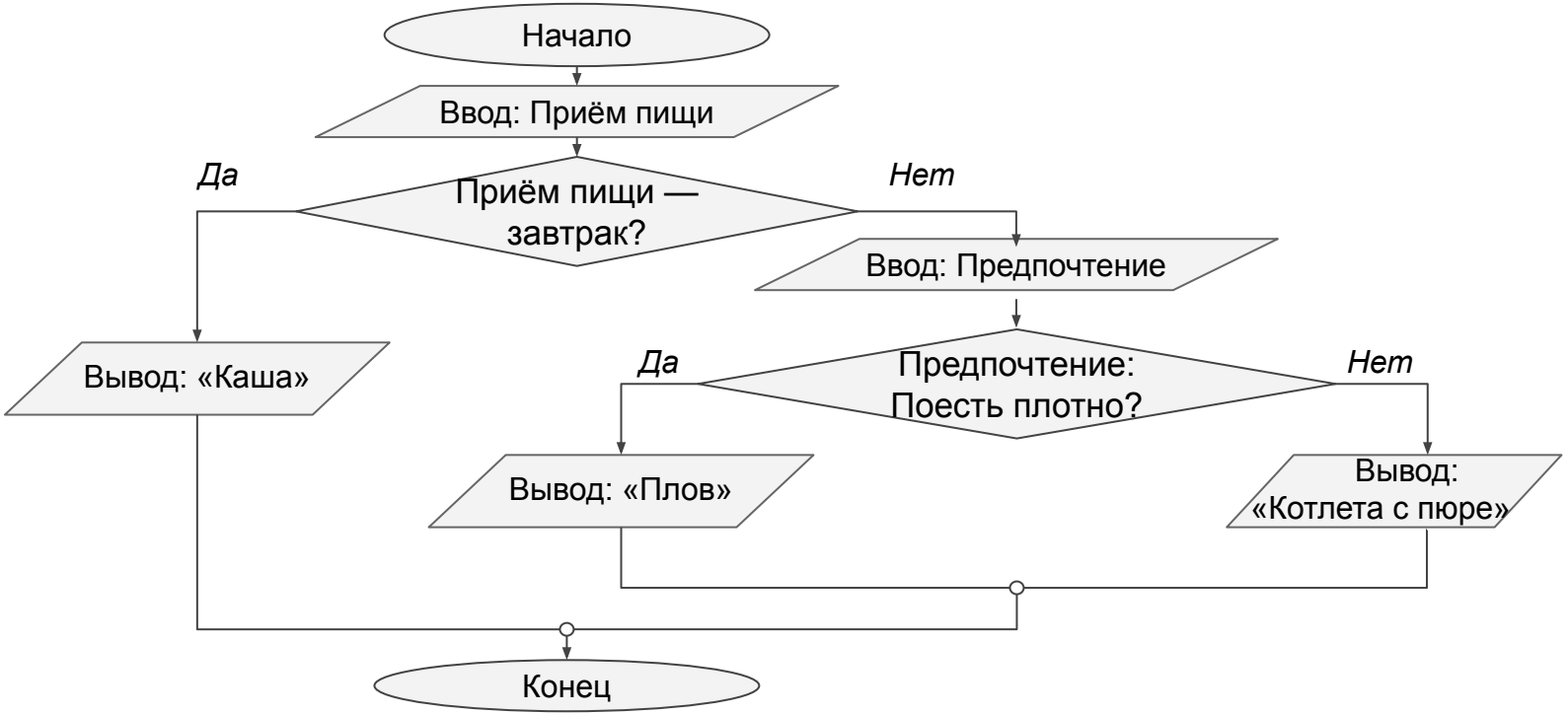
Новых правил оформления нет. Нужно очень внимательно придерживаться уже известных.

Вложенный условный оператор

Задача. Написать программу, предлагающую готовую еду под приём пищи. Если клиент вводит «Завтрак», то рекомендовать кашу. Для других приёмов пищи, если клиент хочет плотно поесть, то рекомендовать плов, в остальных случаях — котлету с пюре.

Вложенный условный оператор

Задача. Написать программу, предлагающую готовую еду под приём пищи. Если клиент вводит «Завтрак», то рекомендовать кашу. Для других приёмов пищи, если клиент хочет плотно поесть, то рекомендовать плов, в остальных случаях — котлету с пюре.



Вложенный условный оператор

Задача. Написать программу, предлагающую готовую еду под приём пищи. Если клиент вводит «Завтрак», то рекомендовать кашу. Для других приёмов пищи, если клиент хочет плотно поесть, то рекомендовать плов, в остальных случаях — котлету с пюре.

```
meal = input('Приём пищи:')
meal = meal.lower()
if meal == 'завтрак':
    print('Каша')
else:
    wish = input('Предпочтение:')
    wish = wish.lower()
    if wish == 'поесть плотно':
        print('Плов')
    else:
        print('Котлета с пюре')
```

```
Приём пищи:
>>> обед
Предпочтение:
>>> поесть плотно
Плов
```

Вложенный условный оператор

Задача. Написать программу, предлагающую готовую еду под приём пищи. Если клиент вводит «Завтрак», то рекомендовать кашу. Для других приёмов пищи, если клиент хочет плотно поесть, то рекомендовать плов, в остальных случаях — котлету с пюре.

```
meal = input('Приём пищи:')
meal = meal.lower()
if meal == 'завтрак':
    print('Каша')
else:
    wish = input('Предпочтение:')
    wish = wish.lower()
    if wish == 'поесть плотно':
        print('Плов')
    else:
        print('Котлета с пюре')
```

Что напечатает программа при последовательном вводе данных:

- Обед — русская кухня
- Обед — поесть плотно
- Ужин — нет предпочтений

Вложенный условный оператор

Задача. Написать программу, предлагающую готовую еду под приём пищи. Если клиент вводит «Завтрак», то рекомендовать кашу. Для других приёмов пищи, если клиент хочет плотно поесть, то рекомендовать плов, в остальных случаях — котлету с пюре.

```
meal = input('Приём пищи:')
meal = meal.lower()
if meal == 'завтрак':
    print('Каша')
else:
    wish = input('Предпочтение:')
    wish = wish.lower()
    if wish == 'поесть плотно':
        print('Плов')
    else:
        print('Котлета с пюре')
```

Что напечатает программа при последовательном вводе данных:

- Завтрак
- Обед — русская кухня
- Обед — поесть плотно
- Ужин — нет предпочтений

Назовите пример данных для ввода, чтобы программа напечатала:

- Котлета с пюре
- Каша
- Плов

Выводы:

1. **Идея вложенных конструкций** применима и для условного оператора. Это помогает избежать лишней сложности проверки условий.
2. Чтобы **запрограммировать** вложенный условный оператор, нужно:
   - определиться с порядком проверки условий;
   - уверенно знать оформление условной конструкции.

Условный оператор с несколькими ветвями

Рассмотрим задачу

Задача «Счастливые часы». Написать программу, запрашивающую ввод суммы к оплате и текущий час. Если товары покупаются с 10 до 12 часов, сумма уменьшается в 2 раза. Если с 20 до 22 часов — в 4 раза. Часы работы «Долголетия»: с 8 до 22.

?

```
Сумма:
>>> 1700
Текущее время (час):
>>> 21
Акция! Итого к оплате: 425.0
```

Как написать такую программу?

Рассмотрим задачу

Задача «Счастливые часы». Написать программу, запрашивающую ввод суммы к оплате и текущий час. Если товары покупаются с 10 до 12 часов, сумма уменьшается в 2 раза. Если с 20 до 22 часов — в 4 раза. Часы работы «Долголетия»: с 8 до 22.

```
total = int(input('Сумма:'))
time = int(input('Текущее время (час):'))
if time >= 10 and time <= 12:
   total = total/2
   print('Акция! Итого к оплате:', total)
if time >= 20 and time <= 22:
   total = total/4
   print('Акция! Итого к оплате:', total)
if time > 8 and time < 10 or time > 12 and time < 20:
   print('Итого к оплате:', total)
```

```
Сумма:
>>> 1700
Текущее время (час):
>>> 21
Акция! Итого к оплате: 425.0
```

Рассмотрим задачу

Задача «Счастливые часы». Написать программу, запрашивающую ввод суммы к оплате и текущий час. Если товары покупаются с 10 до 12 часов, сумма уменьшается в 2 раза. Если с 20 до 22 часов — в 4 раза. Часы работы «Долголетия»: с 8 до 22.

```
total = int(input('Сумма:'))
time = int(input('Текущее время (час):'))
if time >= 10 and time <= 12:
    total = total/2
    print('Акция! Итого к оплате:', total)
if time >= 20 and time <= 22:
    total = total/4
    print('Акция! Итого к оплате:', total)
if time > 8 and time < 10 or time > 12 and time < 20:
    print('Итого к оплате:', total)
```

Иногда объединение частей условия в выражение становится сложным.

Есть ли более простой способ описать все оставшиеся случаи?

Условный оператор нескольких ветвей

Как в случае с вложенным условным оператором, данная конструкция может быть заменена составными выражениями. Но её использование может сделать код проще и оптимальнее.

Если Выражение_1 истинно,

То выполнить действие 1.

Иначе если Выражение_2 истинно,

То выполнить действие 2.

Во всех остальных случаях

Выполнить действие 3.

Рассмотрим задачу

Задача «Счастливые часы». Написать программу, запрашивающую ввод суммы к оплате и текущий час. Если товары покупаются с 10 до 12 часов, сумма уменьшается в 2 раза. Если с 20 до 22 часов — в 4 раза.

```
total = int(input('Сумма:'))
time = int(input('Текущее время (час):'))
if time >= 10 and time <= 12:
   total = total/2
   print('Акция! Итого к оплате:', total)
elif time >= 20 and time <= 22:
   total = total/4
   print('Акция! Итого к оплате:', total)
else:
   print('Итого к оплате:', total)
```

```
Сумма:
>>> 1700
Текущее время (час):
>>> 21
Акция! Итого к оплате: 425.0
```

Рассмотрим задачу

Задача «Счастливые часы». Написать программу, запрашивающую ввод суммы к оплате и текущий час. Если товары покупаются с 10 до 12 часов, сумма уменьшается в 2 раза. Если с 20 до 22 часов — в 4 раза.

```
total = int(input('Сумма:'))
time = int(input('Текущее время (час):'))
if time >= 10 and time <= 12:
   total = total/2
   print('Акция! Итого к оплате:', total)
elif time >= 20 and time <= 22:
   total = total/4
   print('Акция! Итого к оплате:', total)
else:
   print('Итого к оплате:', total)
```

Что напечатает программа при последовательном вводе данных:

- 1000 — 9
- 1000 — 12
- 1000 — 15
- 1000 — 22
- 1000 — 23

Рассмотрим задачу

Задача «Приём пищи». Написать программу, предлагающую еду под приём пищи. Если пользователь вводит «Завтрак», то предлагать «Каша». Если «Обед», то «Суп с тефтелями». Во всех остальных случаях — «Блины с рыбой».

?

```
Приём пищи:
>>> Полдник
Блины с рыбой
```

Как написать такую программу?

Рассмотрим задачу

Задача «Приём пищи». Написать программу, предлагающую еду под приём пищи. Если пользователь вводит «Завтрак», то предлагать «Каша». Если «Обед», то «Суп с тефтелями». Во всех остальных случаях — «Блины с рыбой».

```
meal = input('Приём пищи:')
if meal == 'Завтрак':
    print('Каша')
if meal == 'Обед':
    print('Суп с тефтелями')
if meal != 'Завтрак' and meal != 'Обед':
    print('Блины с рыбой')
```

```
Приём пищи:
>>> Полдник
Блины с рыбой
```

Способ 1. Через составное логическое выражение

Рассмотрим задачу

Задача «Приём пищи». Написать программу, предлагающую еду под приём пищи. Если пользователь вводит «Завтрак», то предлагать «Каша». Если «Обед», то «Суп с тефтелями». Во всех остальных случаях — «Блины с рыбой».

```
meal = input('Приём пищи:')
if meal == 'Завтрак':
   print('Каша')
elif meal == 'Обед':
   print('Суп с тефтелями')
else:
   print('Блины с рыбой')
```

```
Приём пищи:
>>> Полдник
Блины с рыбой
```

Способ 2. Через условный оператор с несколькими ветвями

Рассмотрим задачу

Задача «Приём пищи». Написать программу, предлагающую еду под приём пищи. Если пользователь вводит «Завтрак», то предлагать «Каша». Если «Обед», то «Суп с тефтелями». Во всех остальных случаях — «Блины с рыбой».

```
meal = input('Приём пищи:')
if meal == 'Завтрак':
    print('Каша')
elif meal == 'Обед':
    print('Суп с тефтелями')
else:
    print('Блины с рыбой')
```

Что напечатает программа при вводе данных:

- Завтрак
- Ланч
- Обед
- Не знаю

Способ 2. Через условный оператор с несколькими ветвями

Решаем задачу

Покупатель ищет товары из категории "Праздник". Для этого:
1. Пользователь вводит подкатегорию (например, еда).
2. Если подкатегория — "еда", то программа выводит: "Вам пригодятся: торт, чипсы, газировка".
3. Если подкатегория — "оформление", то программа выводит: "Вам пригодятся: свечи, шары".
4. Во всех остальных случаях — "Загляните в хиты продаж!"

```
Введите подкатегорию:
>>> оформление
Вам пригодятся: свечи, шары
```

Решаем задачу (повышенный уровень)

Программа должна:
1) Запрашивать число покупателей за позавчера и вчера.
2) Сравнивать эти числа.
Если количество растёт, то число покупателей на сегодня это: -> (покупатели вчера + разница за вчера и позавчера). Если количество убывает, то число покупателей на сегодня это: -> (покупатели вчера - разница за вчера и позавчера).
3) Печатать прогноз как на картинке.

```
Введите число покупателей за позавчера:
>>> 200
Введите число покупателей за вчера:
>>> 250
Сегодня магазин посетит: 300 человек
```

Выводы:

1. **Способов** запрограммировать условную конструкцию **много**:
   - условный оператор;
   - вложенный условный оператор;
   - условный оператор с несколькими ветвями.

1. Любую задачу можно решить через «стандартный» условный оператор. Но:
   - **вложенность позволяет сократить** проверку условий;
   - **несколько ветвей упрощают** проверку оставшихся условий.

Для завершения работы пройдите техническое интервью

1. Что такое вложенный условный оператор? Где и зачем он может применяться?
2. Что такое условный оператор с несколькими ветвями? Где и зачем он может применяться?
3. Какой условный оператор лучше: стандартный, вложенный или с несколькими ветвями?

Сегодня вы:

1. Узнали, что вложенный условный оператор — это средство программирования сложных условий.
2. Узнали новые служебные слова if–elif–else для программирования условного оператора с несколькими ветвями.
3. Решали задачи разными способами, выбирая оптимальный.