Условный оператор

Задача «Умные рекомендации»

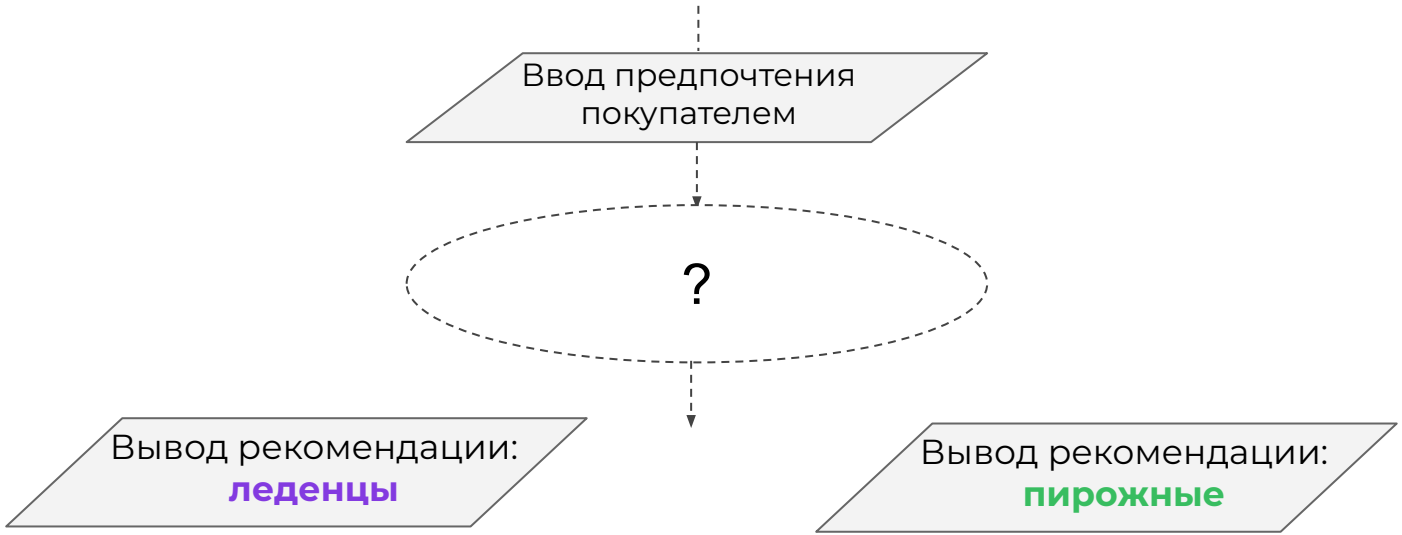
Заказ от кондитерской «Сладкие истории».

Директор кондитерской хочет настроить на сайте **умные рекомендации** товаров.

Умные рекомендации

Упрощенная задача.

Программа знает две рекомендации: **леденцы** и **пирожные**. Пользователь вводит предпочтение: **конфеты**. Как настроить умные рекомендации?

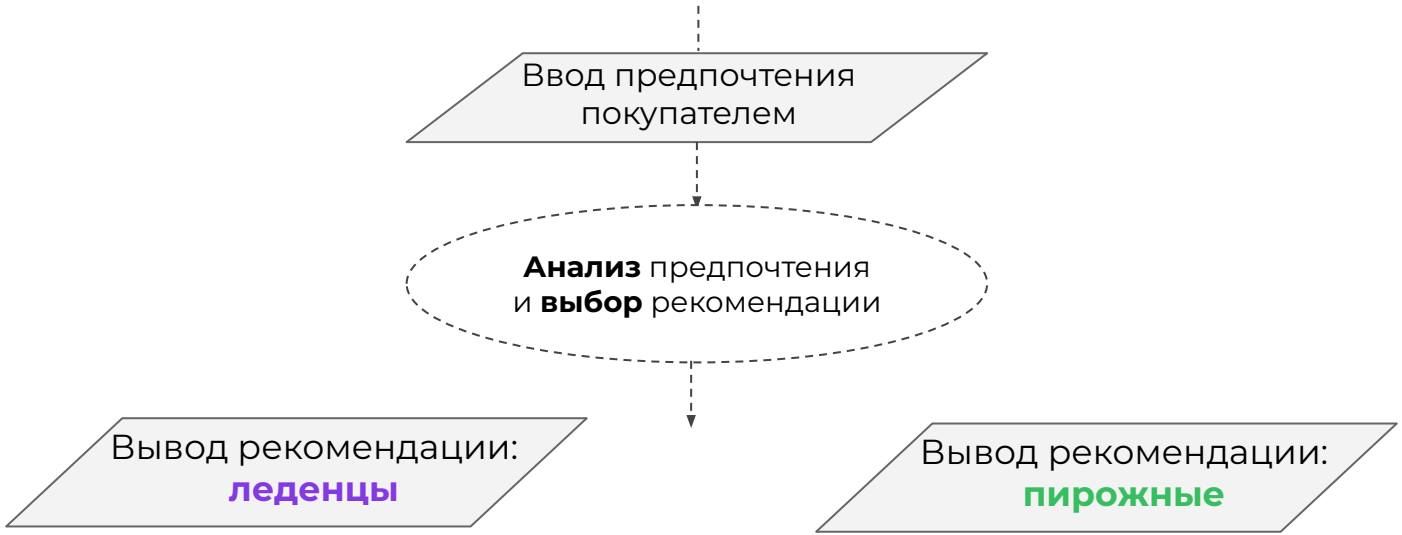


По правилу порядка тут должна быть какая-то команда. Какая?

Умные рекомендации

Упрощенная задача.

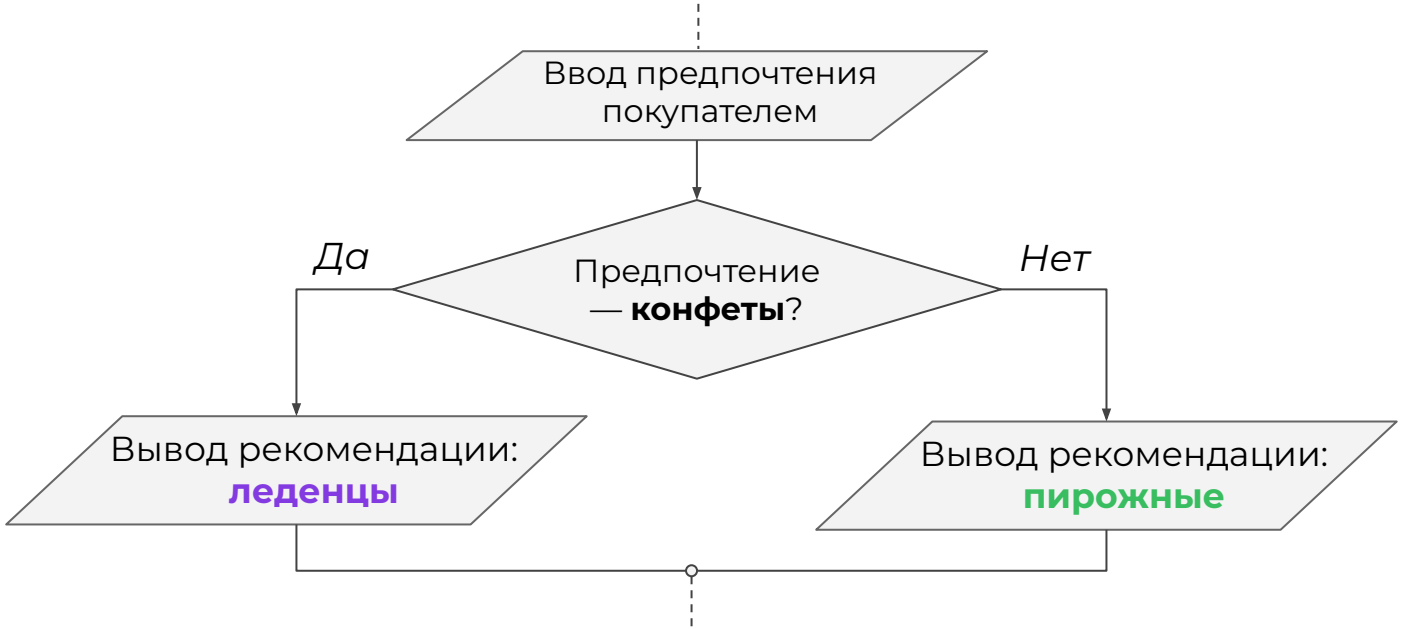
Программа знает две рекомендации: **леденцы** и **пирожные**. Пользователь вводит предпочтение: **конфеты**. Как настроить умные рекомендации?



Умные рекомендации

Упрощенная задача.

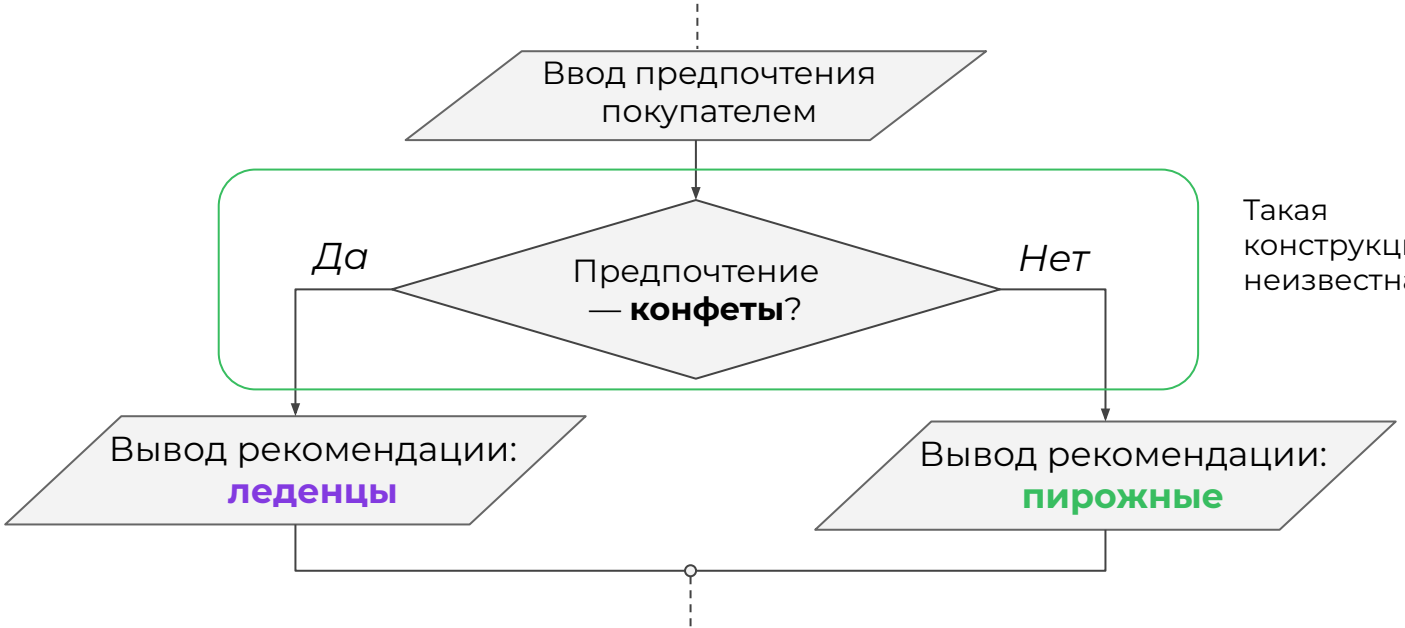
Программа знает две рекомендации: **леденцы** и **пирожные**. Пользователь вводит предпочтение: **конфеты**. Как настроить умные рекомендации?



Умные рекомендации

Упрощенная задача.

Программа знает две рекомендации: **леденцы** и **пирожные**. Пользователь вводит предпочтение: **конфеты**. Как настроить умные рекомендации?



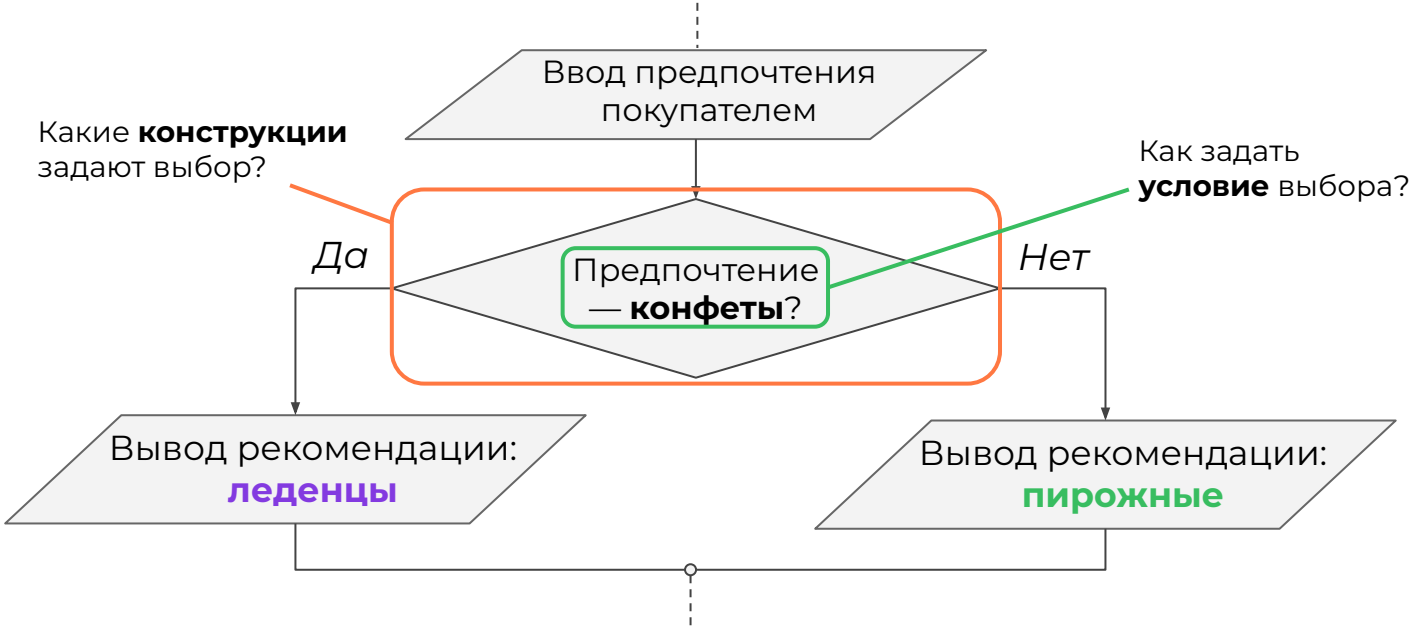
Такая конструкция нам неизвестна...

Что нужно узнать, чтобы запрограммировать такую конструкцию?

Умные рекомендации

Упрощенная задача.

Программа знает две рекомендации: **леденцы** и **пирожные**. Пользователь вводит предпочтение: **конфеты**. Как настроить умные рекомендации?



Сегодня вы:

- ➔ узнаете, что условный оператор — это конструкция, анализирующая условие и выбирающая команду для исполнения;
- ➔ узнаете, как запрограммировать условие с помощью нового типа данных.

Новая тема:
Логический тип данных

Как запрограммировать условие?

В прошлой задаче мы рассматривали **условие**, как некоторое предложение, которое может быть **или истинным, или ложным**.

Логический тип данных

Такие предложения играют важную роль в программировании. Для них был используется **логический тип данных**.

Тип данных	*Целочисленный*	*Логический*
Величины	`-100, 5, 512`	`True, False`
Переменные	`days = 31`	`is_correct = True`
Простые выражения	`daily_money * days` `price - sale`	`5 > 2` `name != 'Иван'`

Переменные и простые выражения

Переменные и выражения могут принимать значение

`True` или `False`.

```
checked = True
is_sent = False
print(checked)
print(is_sent)
```

```
True
False
```

Переменные и простые выражения

Переменные и выражения могут принимать значение

`True` или `False`.

```
checked = True
is_sent = False
print(checked)
print(is_sent)
```

```
True
False
```

```
amount_shop = int(input('Наличие:'))
booked = int(input('Куплено:'))
ok = amount_shop > booked
print(ok)
```

```
Наличие:
>>> 150
Куплено:
>>> 114
True
```

Переменные и простые выражения

Переменные и выражения могут принимать значение

`True` или `False`.

```
checked = True
is_sent = False
print(checked)
print(is_sent)
```

```
True
False
```

```
amount_shop = int(input('Наличие:'))
booked = int(input('Куплено:'))
ok = amount_shop > booked
print(ok)
```

```
Наличие:
>>> 150
Куплено:
>>> 114
True
```

Логическая операция

Логическое выражение

Простое логическое выражение: операторы сравнения

При составлении логических выражений могут использоваться операторы сравнения.

Целочисленный тип					
*	/	%	//	+	-
Умножение	Деление	Остаток от деления	Целая часть от деления	Сложение	Вычитание

Простое логическое выражение: операторы сравнения

При составлении логических выражений могут использоваться операторы сравнения.

Целочисленный тип					
*	/	%	//	+	-
Умножение	Деление	Остаток от деления	Целая часть от деления	Сложение	Вычитание

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

Простое логическое выражение: операторы сравнения

Задача. Написать программу, запрашивающую остаток шоколадных конфет на складе и определяющую, нужно ли пополнить хранилище. Минимально допустимое количество конфет на складе — 50 кг.

Возможно, необходимость доставки конфет можно задать с помощью логического выражения.

Простое логическое выражение:
операторы сравнения

Задача. Написать программу, запрашивающую остаток шоколадных конфет на складе и определяющую, нужно ли пополнить хранилище. Минимально допустимое количество конфет на складе — 50 кг.

```
amount_store = int(input('Наличие:'))
amount_min = 50
delivery = amount_store < amount_min
print('Нужна доставка:', delivery)
```

```
Наличие:
>>> 50
Нужна доставка: False
```

```
Наличие:
>>> 49
Нужна доставка: True
```

Составное логическое выражение

Составное логическое выражение можно создать **из простых** выражений, 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.*

Составное логическое выражение

Задача. Написать программу, оповещающую об ошибке хранения в хранилище шоколадных конфет.

Ошибка хранения возникает, когда хранилище почти опустело (меньше 50 кг) или когда оно переполнено (больше 300 кг).

Попробуйте запрограммировать ошибку хранения с помощью составного логического выражения

Составное логическое выражение

Задача. Написать программу, оповещающую об ошибке хранения в хранилище шоколадных конфет.

Ошибка хранения возникает, когда хранилище почти опустело (меньше 50 кг) или когда оно переполнено (больше 300 кг).

```
amount_store = int(input('Наличие:'))
error = amount_store < 50 or amount_store > 300
print('Ошибка хранения:', error)
```

```
Наличие:
>>> 540
Ошибка хранения: True
```

```
Наличие:
>>> 275
Ошибка хранения: False
```

Составное логическое выражение

Задача. Написать программу, оповещающую об ошибке хранения в хранилище шоколадных конфет.

Ошибка хранения возникает, когда хранилище почти опустело (меньше 50 кг) или когда оно переполнено (больше 300 кг).

```
amount_store = int(input('Наличие:'))
error = amount_store < 50 or amount_store > 300
print('Ошибка хранения:', error)
```

Сначала вычисляются значения простых выражений, затем — составного выражения.

```
Наличие:
>>> 540
Ошибка хранения: True
```

```
Наличие:
>>> 275
Ошибка хранения: False
```

Выводы:

1. Логический тип данных — это тип для программирования выражений, которые могут быть истинными или ложными.
2. Простые логические выражения можно создать с помощью операторов сравнения.
3. Составные логические выражения можно создать из простых логических выражений и логических операторов.

== !=		Логический оператор И
and		Логический оператор ИЛИ
< <=		Оператор сравнения БОЛЬШЕ, БОЛЬШЕ ИЛИ РАВНО
or		Операторы сравнения МЕНЬШЕ, МЕНЬШЕ ИЛИ РАВНО
> >=		Операторы сравнения РАВНО, НЕ РАВНО

and	Логический оператор И
or	Логический оператор ИЛИ
> >=	Оператор сравнения БОЛЬШЕ, БОЛЬШЕ ИЛИ РАВНО
< <=	Операторы сравнения МЕНЬШЕ, МЕНЬШЕ ИЛИ РАВНО
== !=	Операторы сравнения РАВНО, НЕ РАВНО

```
login = 'ivanova.ekaterina'
password = input(login + ', введите пароль для входа в личный кабинет:’)

print('Авторизация:’,  )
```

Чтобы получить персональные рекомендации, покупатель должен войти в личный кабинет. Пользователь ivanova.ekaterina заявила об ошибке входа. Она вводит свой пароль sweet111, но войти в кабинет не удаётся. Исправь ошибки в авторизации пользователя. Программа должна срабатывать как на картинке.

```
ivanova.ekaterina, введите пароль для входа в личный кабинет:
>>> sweet111
Авторизация: True
```

Некоторые покупатели предпочитают диетические сладости. К ним относятся пожелания:
- "без сахара";
- "0% жирности";
- "без глютена".

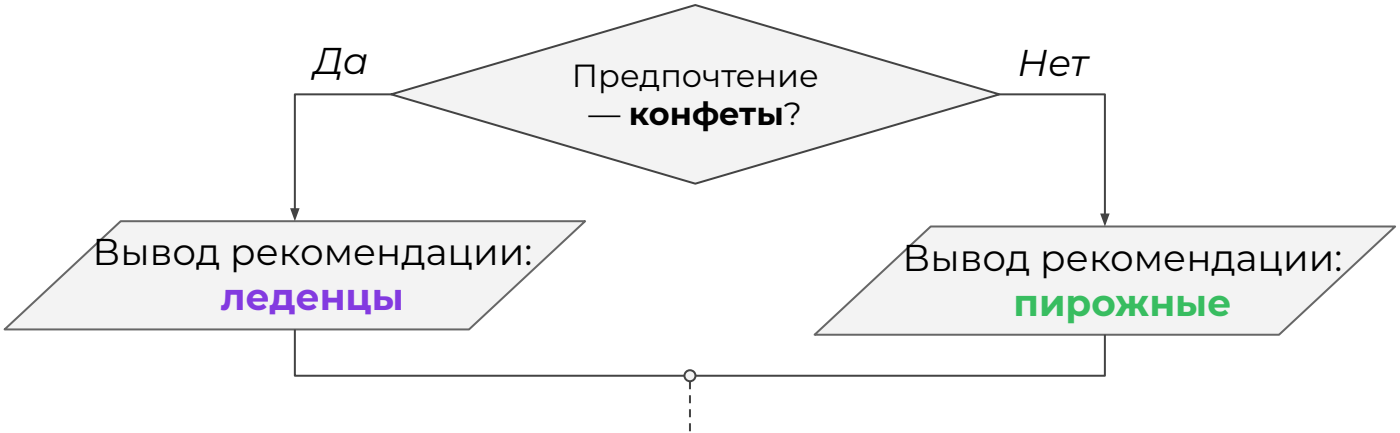
Достаточно упоминания одного такого пожелания. Напиши программу, определяющую, нужны ли покупателю диетические продукты. Результат работы должен быть как на картинке.

```
Пожелание:
>>> без сахара
Предлагать диетические товары: True
```

Как запрограммировать выбор?

Мы узнали, как запрограммировать условие — предложение, которое может быть истинным или ложным.

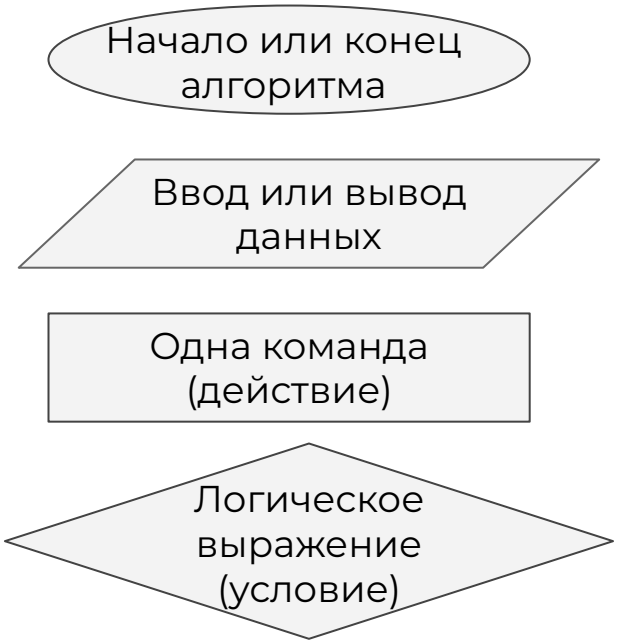
Теперь освоим конструкцию, осуществляющую выбор команды для выполнения, в зависимости от истинности условия.



Запись алгоритма в виде блок-схемы

Здесь и далее при разборе алгоритмических конструкций мы будем использовать блок-схемы.

Это универсальный способ записи алгоритма, который известен каждому программисту.

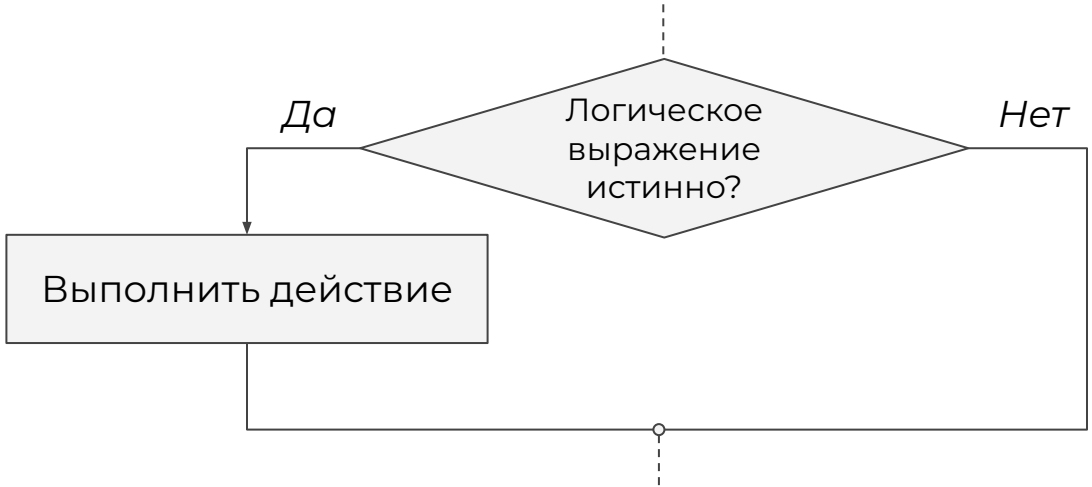


Виды блоков

Условный оператор

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

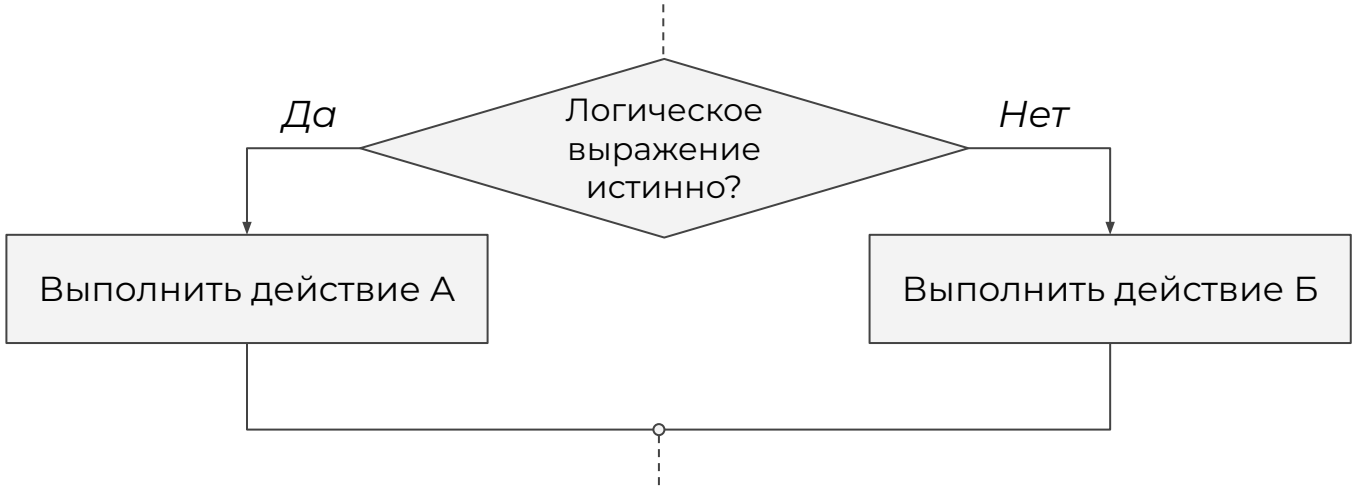
Пример использования:
выполнение некоторого действия только если выражение истинно.



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.



Условный оператор

Задача 1. Составить алгоритм, проверяющий возможность покупки по карте.

Если стоимость товаров больше, чем сумма на карте, то вывести:
«Недостаточно средств».

Условный оператор

Задача 1. Составить алгоритм, проверяющий возможность покупки по карте.

Если стоимость товаров больше, чем сумма на карте, то вывести:
«Недостаточно средств».

Условный оператор

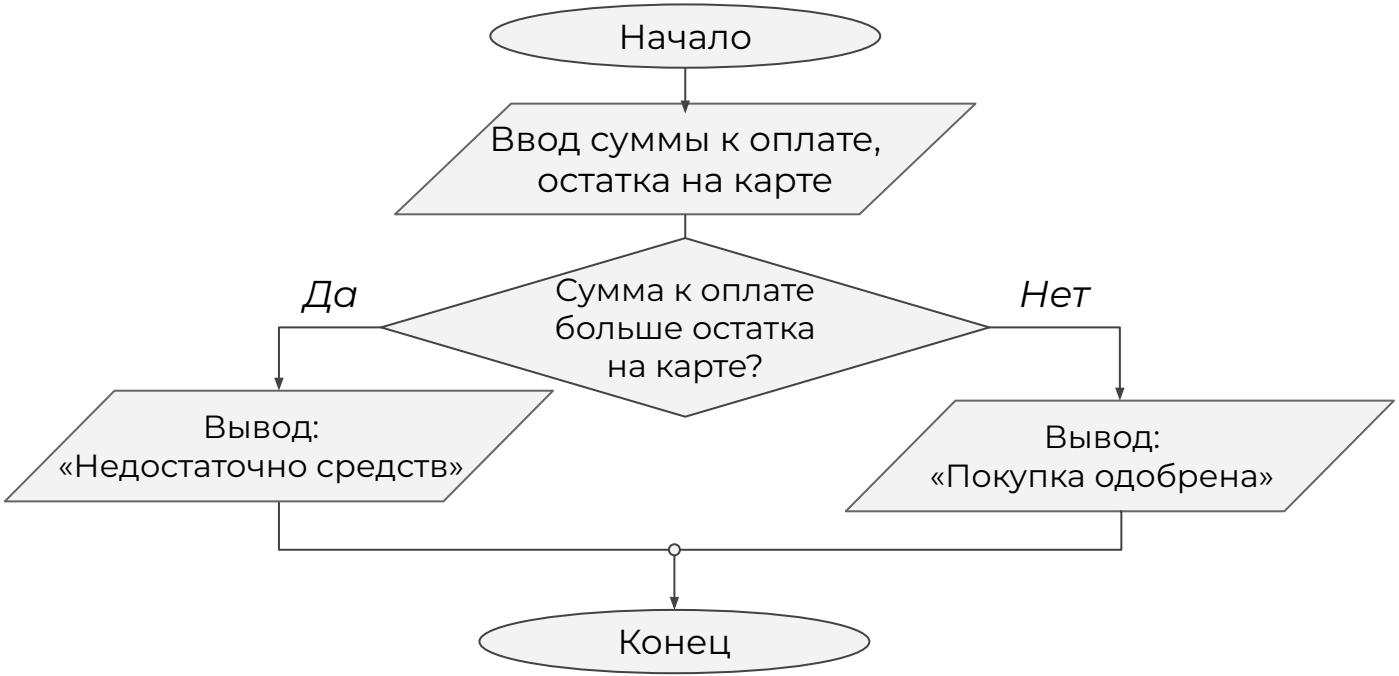
Задача 2. Составить алгоритм проверяющий возможность покупки по карте.

Если стоимость товаров больше, чем сумма на карте, то вывести:
«Недостаточно средств». Иначе — вывести «Покупка одобрена».

Условный оператор

Задача 2. Составить алгоритм проверяющий возможность покупки по карте.

Если стоимость товаров больше, чем сумма на карте, то вывести:
«Недостаточно средств». Иначе — вывести «Покупка одобрена».



Условный оператор

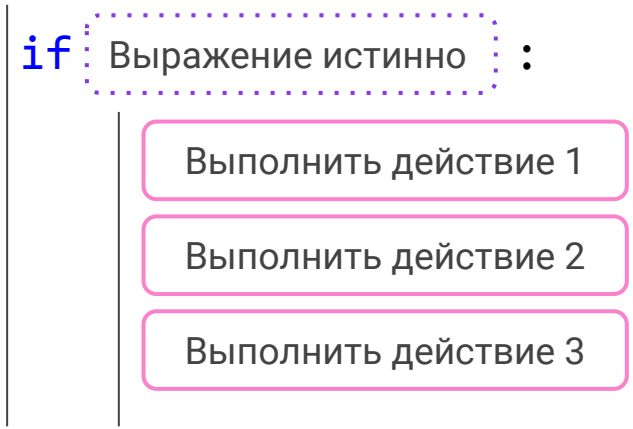
Для программирования условного оператора используются команды:

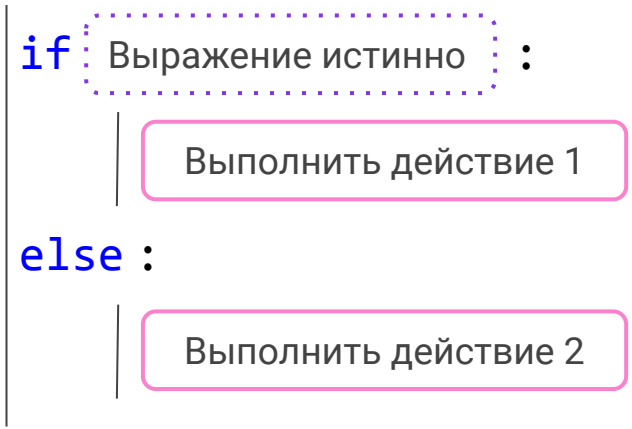
- `if` (*в англ. — «если»*);
- `else` (*в англ. — «иначе»*).

Условный оператор

Для программирования условного оператора используются команды:

- **`if`** (*в англ. — «если»*);
- **`else`** (*в англ. — «иначе»*).



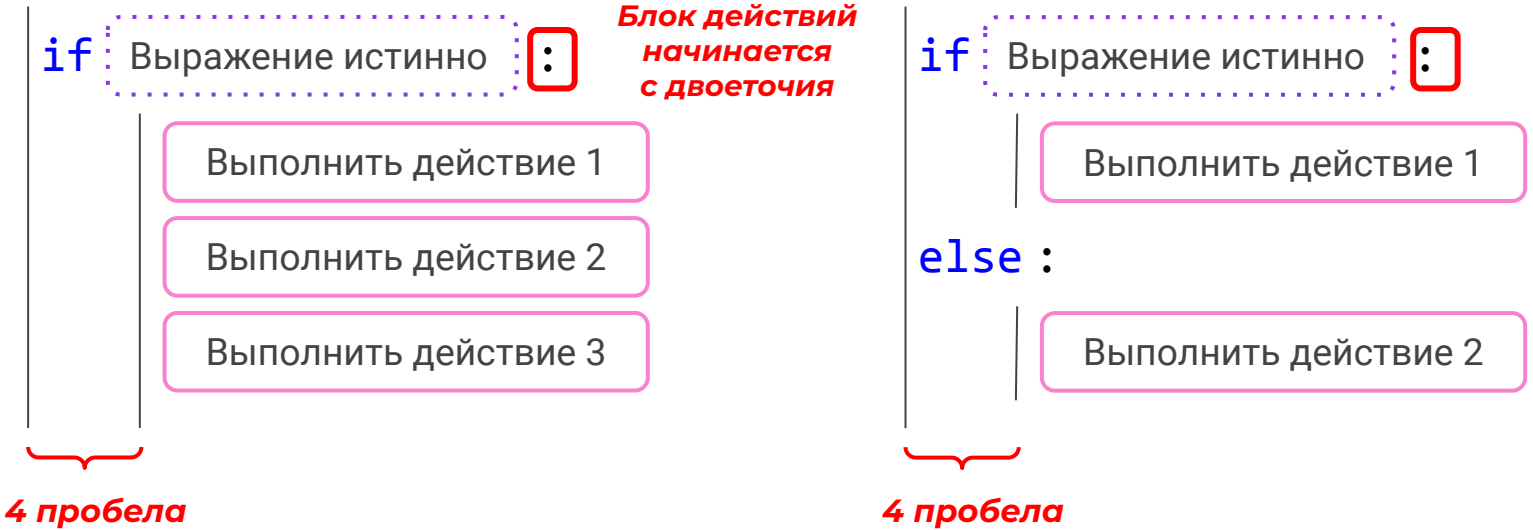


Условный оператор

Для программирования условного оператора используются команды:

`if` (*в англ. — «если»*)

`else` (*в англ. — «иначе»*)

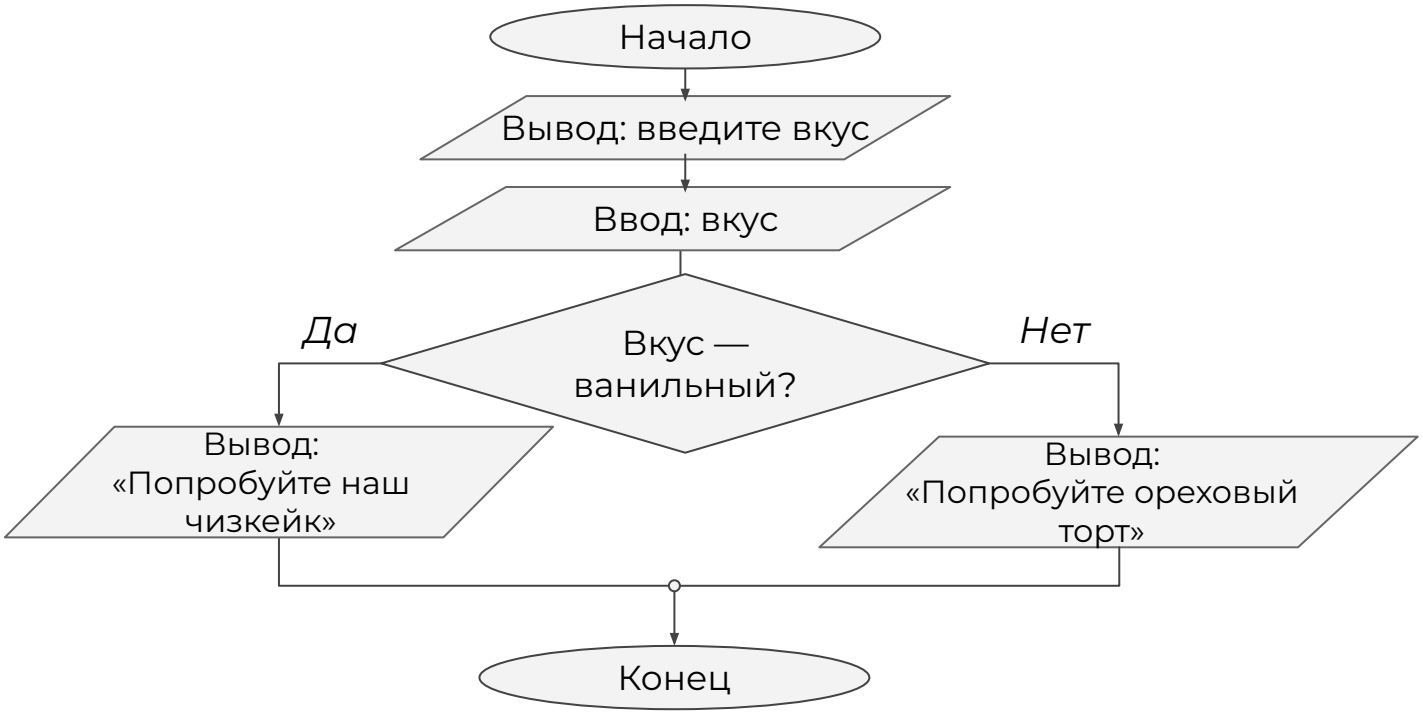


Условный оператор

Задача. Написать программу, предлагающую товар по вкусовым предпочтениям. Программа спрашивает, какой вкус нравится пользователю. Если ванильный, то рекомендовать чизкейк. Иначе — ореховый торт.

Условный оператор

Задача. Написать программу, предлагающую товар по вкусовым предпочтениям. Программа спрашивает, какой вкус нравится пользователю. Если ванильный, то рекомендовать чизкейк. Иначе — ореховый торт.



Условный оператор

Задача. Написать программу, предлагающую товар по вкусовым предпочтениям. Программа спрашивает, какой вкус нравится пользователю. Если ванильный, то рекомендовать чизкейк. Иначе — ореховый торт.

```
taste = input('Введите любимый вкус:')
taste = taste.lower()
```

Условный оператор

Задача. Написать программу, предлагающую товар по вкусовым предпочтениям. Программа спрашивает, какой вкус нравится пользователю. Если ванильный, то рекомендовать чизкейк. Иначе — ореховый торт.

```
taste = input('Введите любимый вкус:')
taste = taste.lower()
if taste == 'ванильный':
   print('Попробуйте наш фирменный чизкейк!')
else:
   print('Попробуйте ореховый торт!')
```

```
Введите любимый вкус:
>>> Ванильный
Попробуйте наш фирменный чизкейк!
```

```
Введите любимый вкус:
>>> шоколадный
Попробуйте ореховый торт!
```

Напишите программу, рекомендующую товары в зависимости от суммы, которую вводит пользователь. Если покупатель готов:
- потратить меньше 500 рублей, то рекомендуются пирожные;
- потратить от 500 до 1000 рублей включительно, то рекомендуется тортик Секрет;
- потратить больше 1000 рублей, то рекомендуется шоколадный фондан.
Возможный результат работы программы представлен на картинке. Вводимая сумма может быть любой!

```
Какую сумму вы готовы потратить на сладости?
>>> 690
Побалуйте себя тортиком Секрет!
```

Выводы:

1. Условный оператор — это команда, выполняющая или не выполняющая действие в зависимости от значения логического выражения.
2. Для программирования условного оператора используются операторы `if` и `else`.
3. Действия внутри условного оператора начинаются с двоеточия и пишутся с отступом в 4 пробела.

Сегодня вы:

1. Узнали тип данных для программирования выражений, принимающих значения истина или ложь.
2. Узнали новую алгоритмическую конструкцию — условный оператор.
3. Научились программировать выбор исполняемой команды в зависимости от истинности условия.