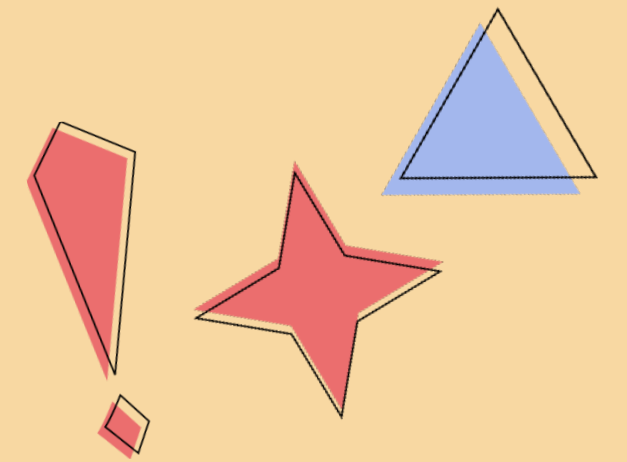


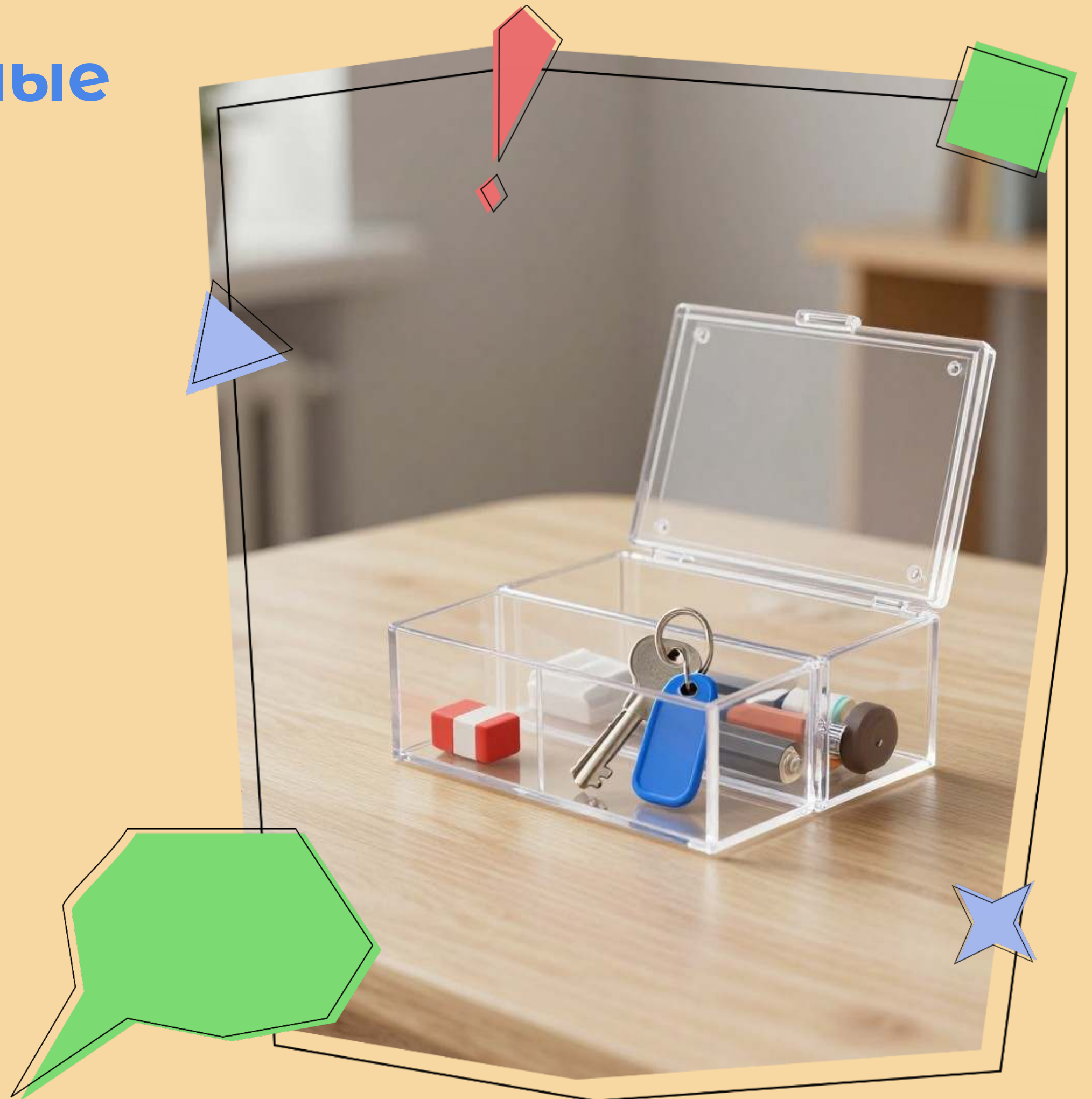
Комбинированный тип данных: структура, назначение и примеры

Комбинированный тип объединяет связанные значения в одной сущности, чтобы описывать объекты реального мира.

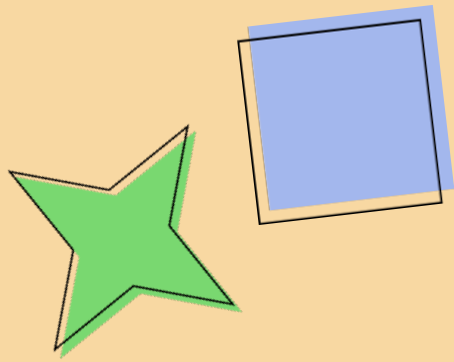


Зачем нужны составные данные в программе

Один объект реальности редко описывается одним числом или строкой. Ученик, книга или товар имеют набор признаков. Раздельное хранение усложняет код и повышает риск ошибок при передаче параметров; составные типы сохраняют смысл связи между полями.

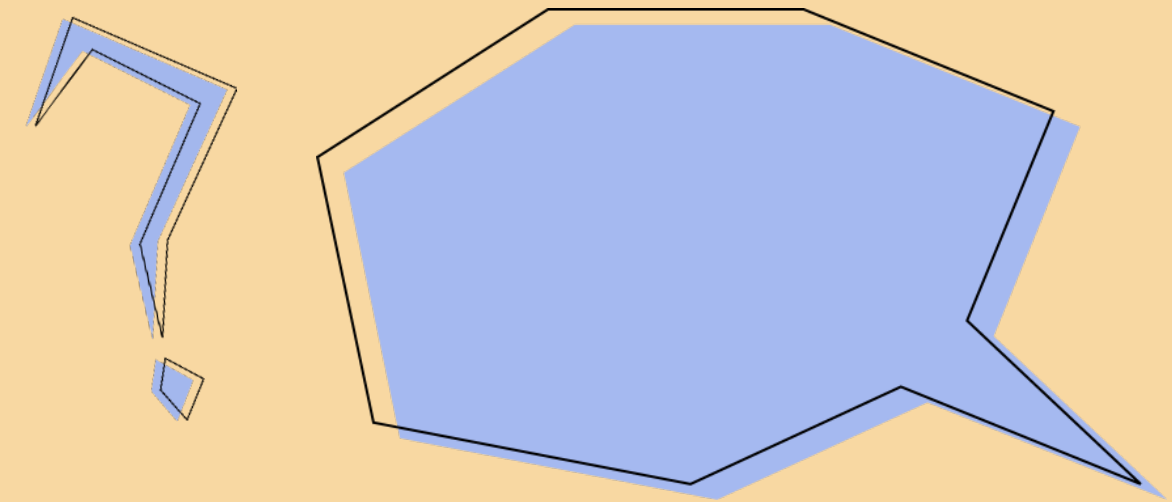


Определение комбинированного типа данных

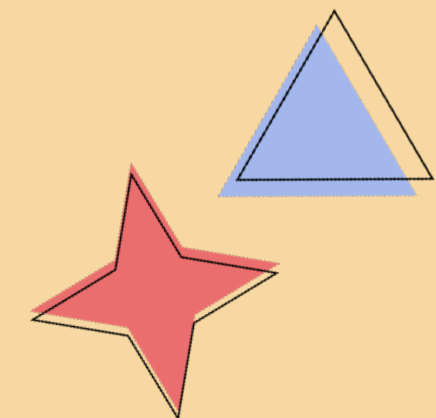


Комбинированный тип данных — это составная структура, которая объединяет несколько элементов под одним именем.

Каждый элемент соответствует отдельному полю и может иметь собственный тип: числовой, строковый, логический или даже другой составной тип.



Такая форма хранения описывает объект целиком, поэтому подходит там, где важно не одно значение, а набор характеристик, тесно связанных между собой.



Примеры объектов, которые удобно описывать структурами



Паспортная карточка

В паспорте важны данные, которые нельзя смешивать: фамилия, номер документа, дата выдачи и организация. Удобно представить это как «одну карточку», где каждое поле хранится отдельно, но все вместе дают полную информацию о человеке.



Карточка координаты

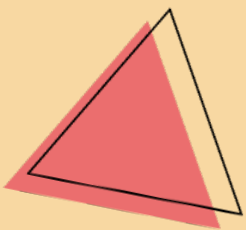
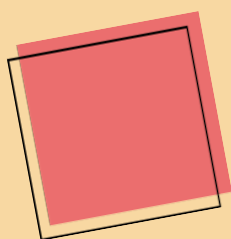
Точка на плоскости — это не просто одно число, а пара значений: x и y . Структура позволяет держать их вместе, чтобы расстояние, перемещение или проверка принадлежности области работали с целой точкой.



Карточка ученика

Ученик — это набор признаков: фамилия, класс, средний балл и даже номер группы. Когда данные собраны в структуру, алгоритмы получают целостный объект и обрабатывают его как единое целое, а не как набор разрозненных переменных.

Простые, массивы и комбинированные типы: различия

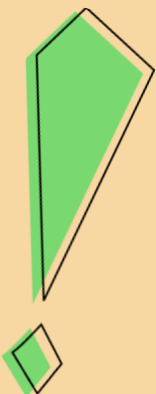
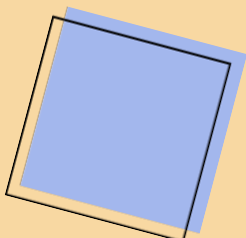
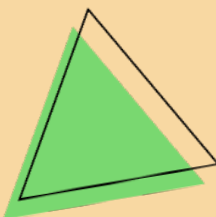


Тип	Что хранит	Особенность
Простой	Одно значение	Например, число
Массив	Много одинаковых элементов	Одного назначения
Комбинированный	Связанные поля разных типов	Именованные признаки объекта
Структура/класс	Данные + (иногда) методы	Инкапсуляция логики

Сравнение по базовым признакам: что именно хранит тип, как устроены его элементы, и чем структуру отличает моделирование целого объекта.

Ключевое отличие: комбинированный тип моделирует объект как целое с полями, а не как последовательность одинаковых элементов.

Базовые понятия информатики и программирования.



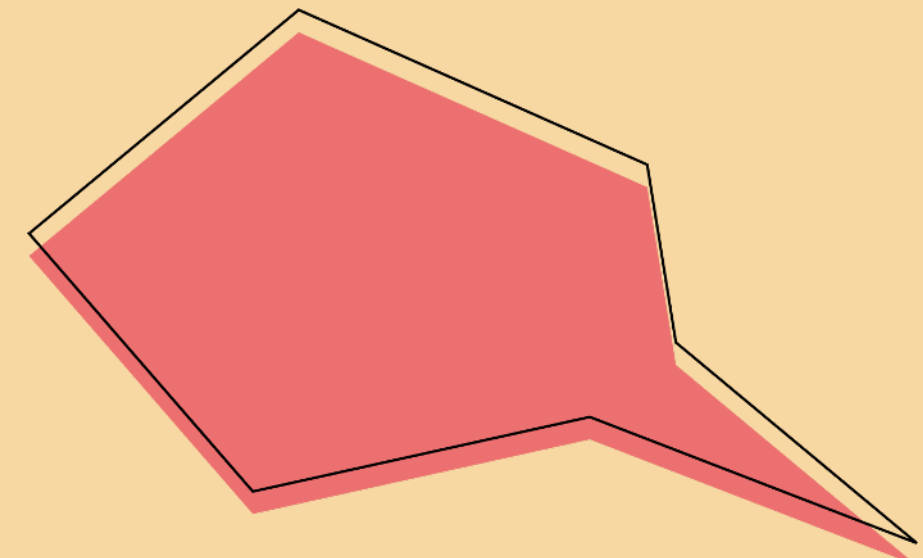
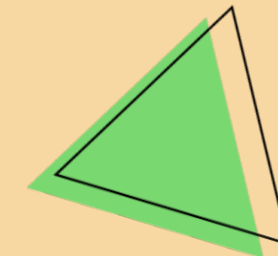
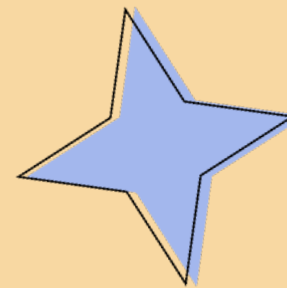
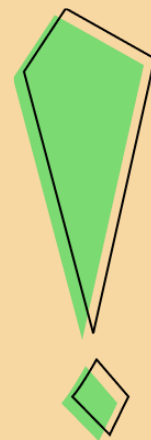
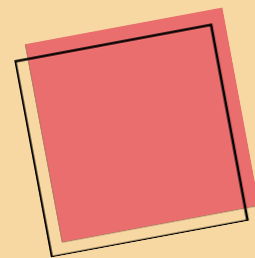
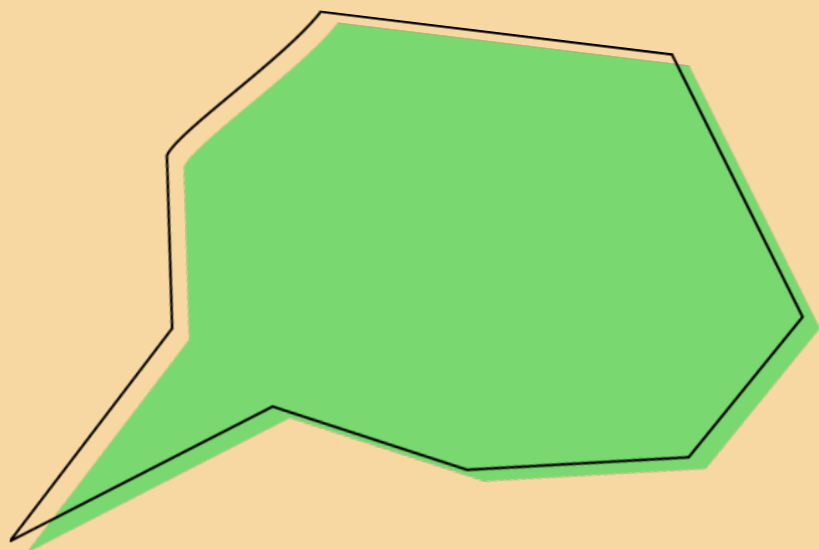
Основные свойства составных данных

Комбинированные типы имеют фиксированный набор полей, поэтому модель данных предсказуема и пригодна для обучения алгоритмам. Это снижает хаотичность и облегчает чтение программы.

У каждого поля есть имя, а значит, к значениям можно обращаться целенаправленно: менять один признак, не трогая другие. Такой подход особенно удобен при отладке.

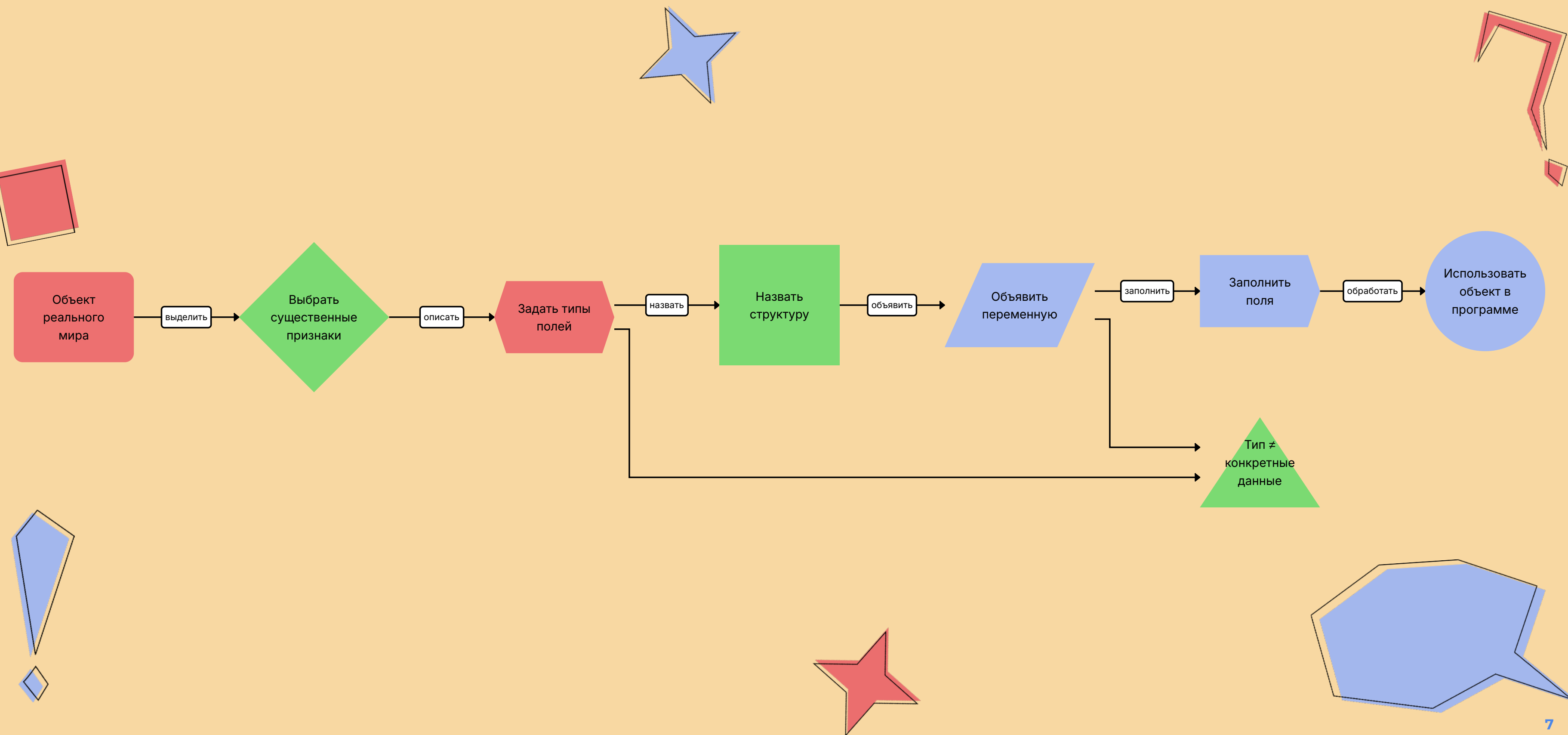
В одной записи могут храниться значения разных типов: строка и число, логическое условие и дата. Поля логически связаны и описывают один объект или событие.

Из-за единой структуры проще копировать объект, передавать его в процедуры и функции и обращаться к конкретным свойствам без разбиения информации на множество независимых переменных.



Как создается структура данных для объекта

Процесс проектирования: от реального объекта к типу и экземпляру с данными.

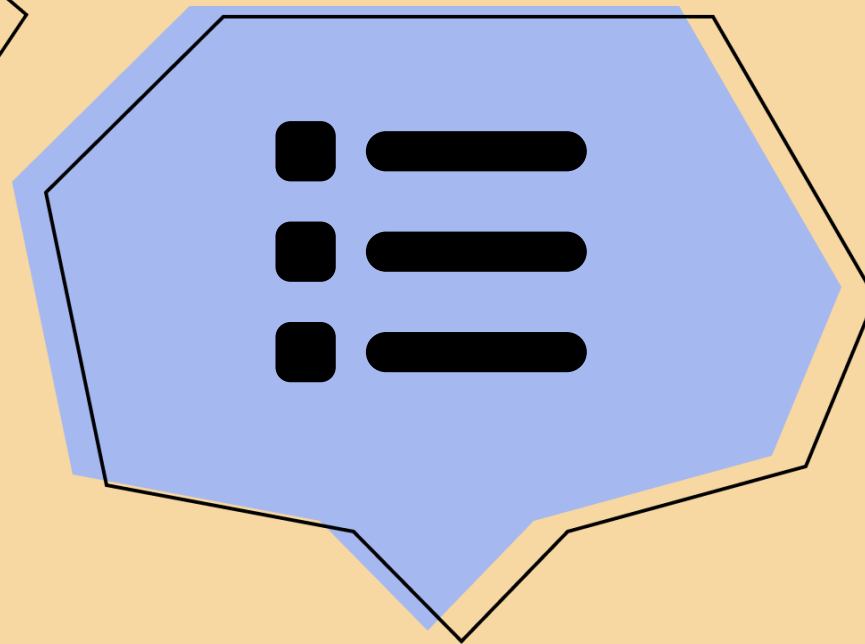
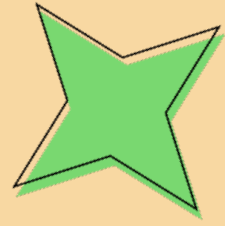


Поля, записи и вложенные элементы



Именованные поля

В комбинированном типе каждое поле имеет имя и свой тип. Благодаря этому программа понимает, где хранится фамилия, цена или статус, и обращения становятся точными и читаемыми.



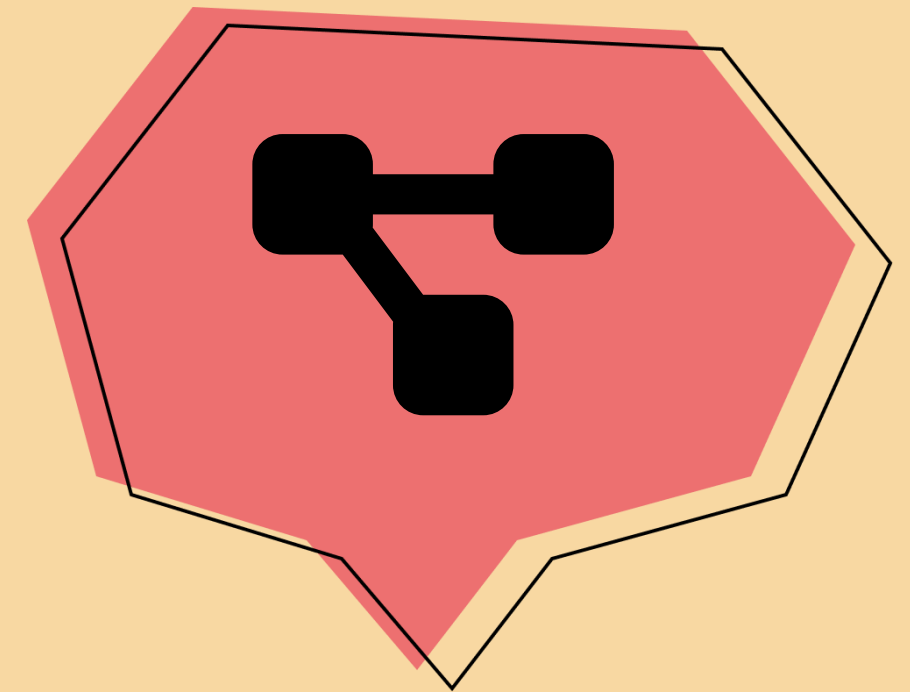
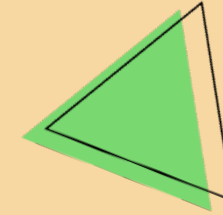
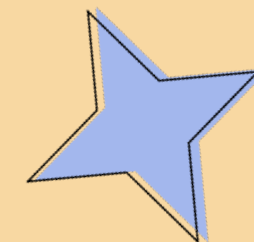
Запись как единая сущность

Запись — это конкретный экземпляр структуры с заполненными значениями полей. В алгоритмах её удобно передавать целиком: параметры не теряются, а смысл данных сохраняется.



Разные типы внутри одной записи

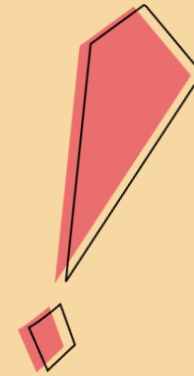
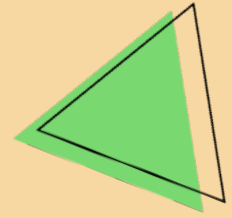
Поле может хранить число, строку, логическое значение, дату или другую структуру. Такой набор позволяет описывать объект не частями, а полноценно, как он существует в реальности.



Вложение для иерархии

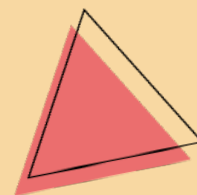
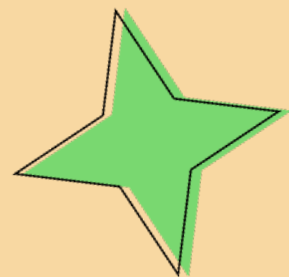
Вложенные структуры делают модель точнее: например, у заказа может быть список товаров, а у человека — паспортные данные. В итоге отражается иерархия информации, а не случайное смешение фактов.

Состав данных в типичной структуре

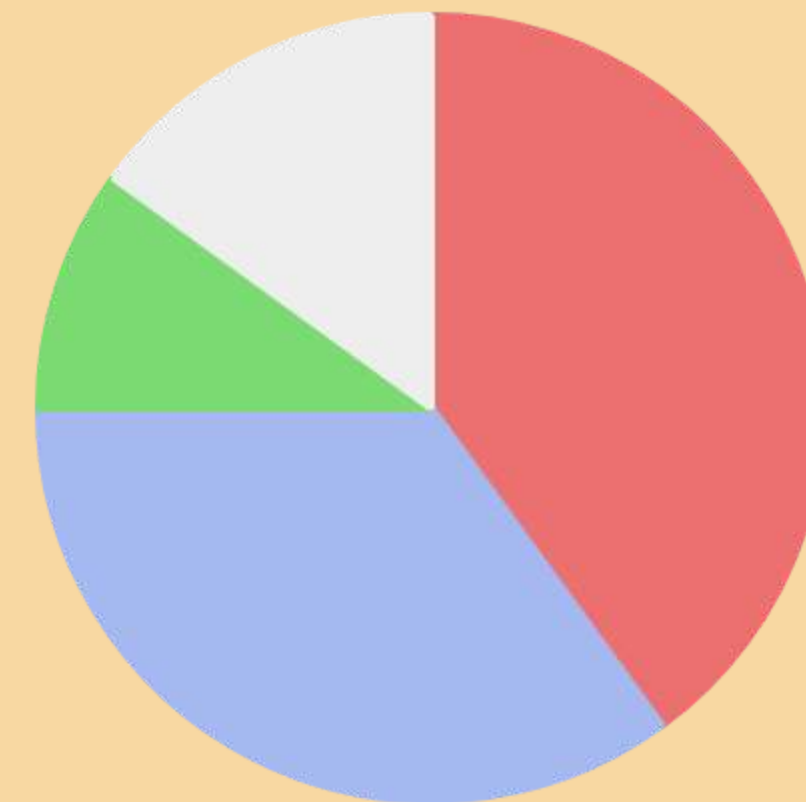


Проценты в учебном примере показывают, что в одной модели часто встречаются разные категории полей, а не повторение одного типа.

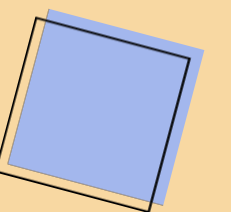
Разнообразие типов делает структуры универсальными для моделирования: текст задаёт идентификаторы и описания, числа — характеристики, логика — условия, вложения — детали.



■ Строковые ■ Числовые ■ Логические
■ Вложенные структуры



Учебная иллюстрация к теме составных данных (распределение задано в постановке примера).

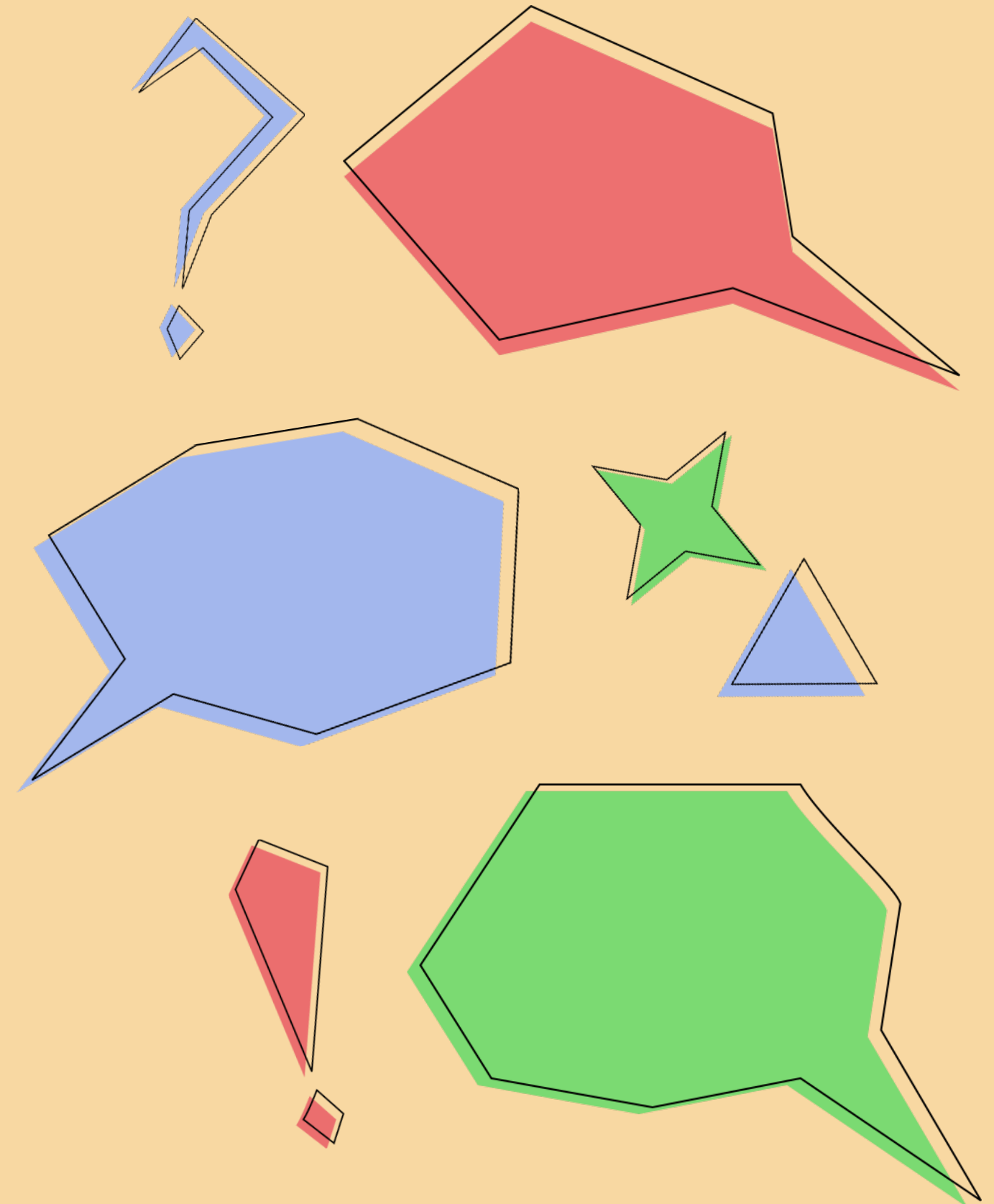
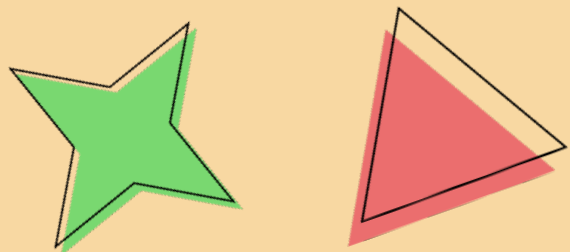


Преимущества использования комбинированных типов

Комбинированные типы уменьшают число отдельных переменных: данные об одном объекте собираются в единую запись, а программа становится более упорядоченной и логичной по структуре.

Объект можно передавать в функцию как целое, снижая риск перепутать параметры с похожими названиями. Это особенно важно при обработке списков карточек.

Когда характеристик много, код обычно получается короче и понятнее: структура данных ближе к тому, как мы описываем объект словами, а значит легче проверять логику.



Ограничения и возможные ошибки



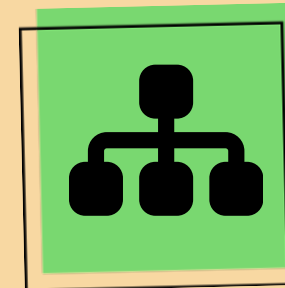
Слишком громоздкая структура

Если полей становится много, модель сложнее читать и сопровождать. Команда разработчиков тратит время на понимание, где чьи данные, как они связаны и какие поля обязательны для алгоритмов и проверок.



Поля без логической связи

Когда в одной записи объединяют разнородные сведения, теряется смысл составного типа. Тогда объект перестаёт описывать одну сущность, а операции чтения и передачи становятся запутанными и ошибкоопасными.



Избыточная вложенность и смешение сущностей

Проблемы возникают при чрезмерном уровне вложения или попытке хранить в одном объекте данные разных сущностей. Это усложняет доступ к полям и провоцирует логические сбои при обновлении значений.

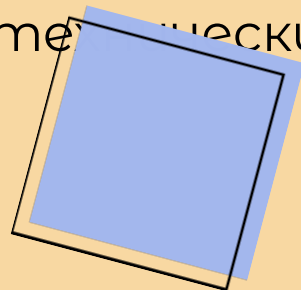
Комбинированные типы в разных языках программирования

Конструкции для описания «карточек» данных

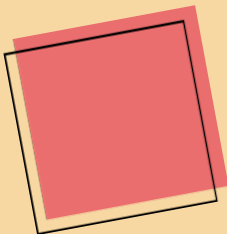
В Pascal для хранения группы связанных полей применяются записи. В C используют структуры: набор полей под одним именем. Эти средства удобны, когда важны единое имя объекта и осмысленные поля.

Как это проявляется в современных языках

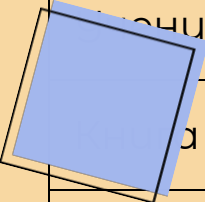
В Python чаще встречаются словари, а также dataclass и классы: они позволяют структурировать данные и задавать правила их использования. В Java применяют классы и record. В итоге составные данные описывают пользователей, товары, документы и тематические объекты.



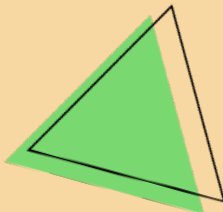
Учебные примеры полей для составных объектов



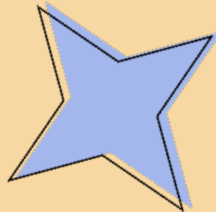
Объект	Типичные поля
Ученик	фамилия; класс; оценки
Книга	название; автор; год
Автомобиль	марка; мощность; пробег
Координата	x; y
Заказ	номер; сумма; статус



Типовые задачи по информатике и программированию

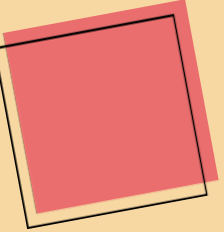
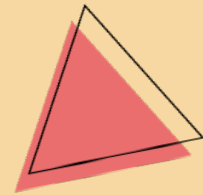


В таблице показано, как типичный объект обычно требует не одного значения, а набора признаков. Вместе они формируют полную характеристику объекта, позволяя выполнять операции: сравнение, поиск, вывод и обработку.



Один объект представляют как набор связанных полей: это помогает сохранить смысловую связь между характеристиками и избежать «рассыпавшихся» параметров по отдельным переменным.

Вложенные структуры как способ точного описания объектов



1. Определить общие сведения

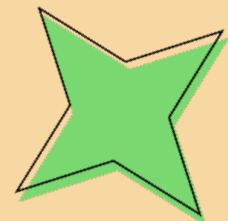
Сначала выделяют то, что относится к объекту в целом: например, у человека часто есть имя и базовые реквизиты. Эти поля удобно держать на верхнем уровне структуры, чтобы быстро обращаться к ним.

2. Вложить детализированные данные

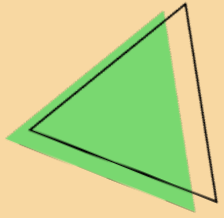
Затем добавляют поля, которые сами являются объектами: у человека — адрес, у заказа — список позиций, у студента — данные о школе и контакты. Вложенность помогает отражать иерархию информации.

3. Проверить глубину и читаемость

На финальном этапе оценивают, не стала ли структура слишком сложной. Чрезмерная вложенность затрудняет чтение кода и повышает риск ошибок, поэтому важно проектировать модель аккуратно и по задаче.

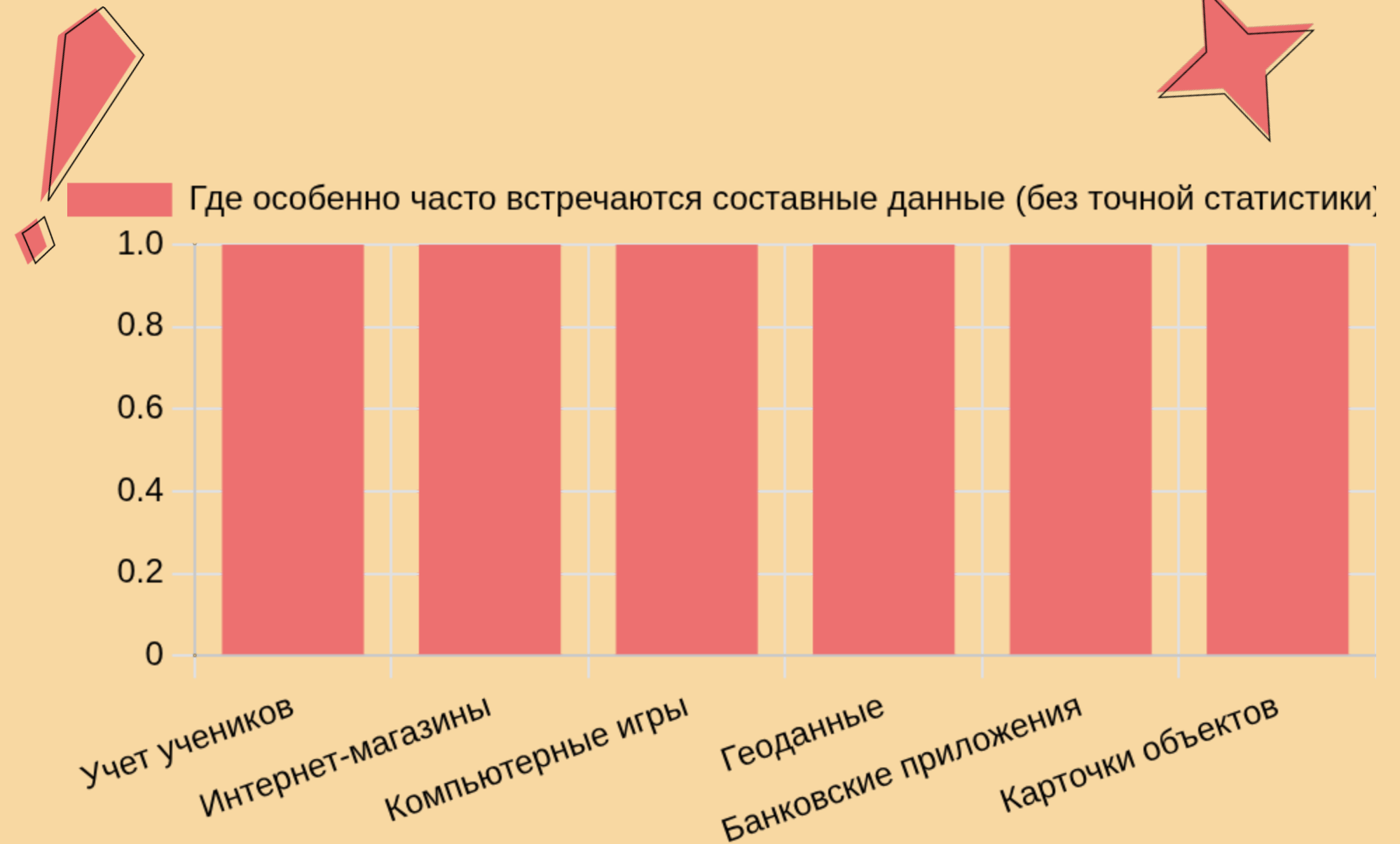
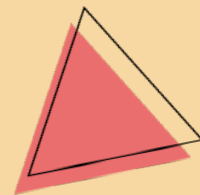
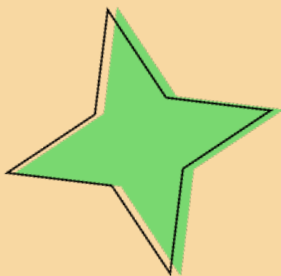


Где комбинированные типы применяются чаще всего

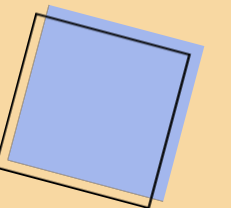


Во всех перечисленных областях важны поля, состояния и связанные сведения. Чаще всего составные данные нужны именно для «карточек» объектов, где параметры логически связаны.

Диаграмма показывает, что потребность в составных данных возникает в самых разных прикладных темах. Главный общий признак — один объект содержит множество взаимосвязанных характеристик.



Обобщение типичных учебных примеров и школьных задач (точные доли не задавались).

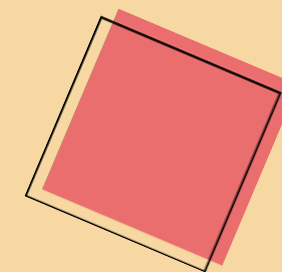
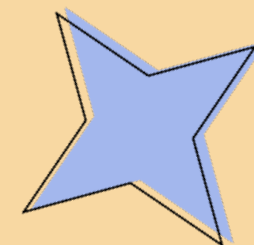
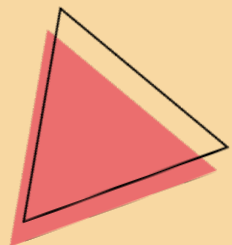
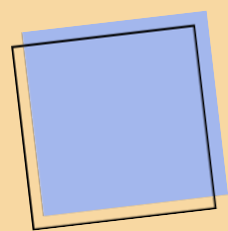
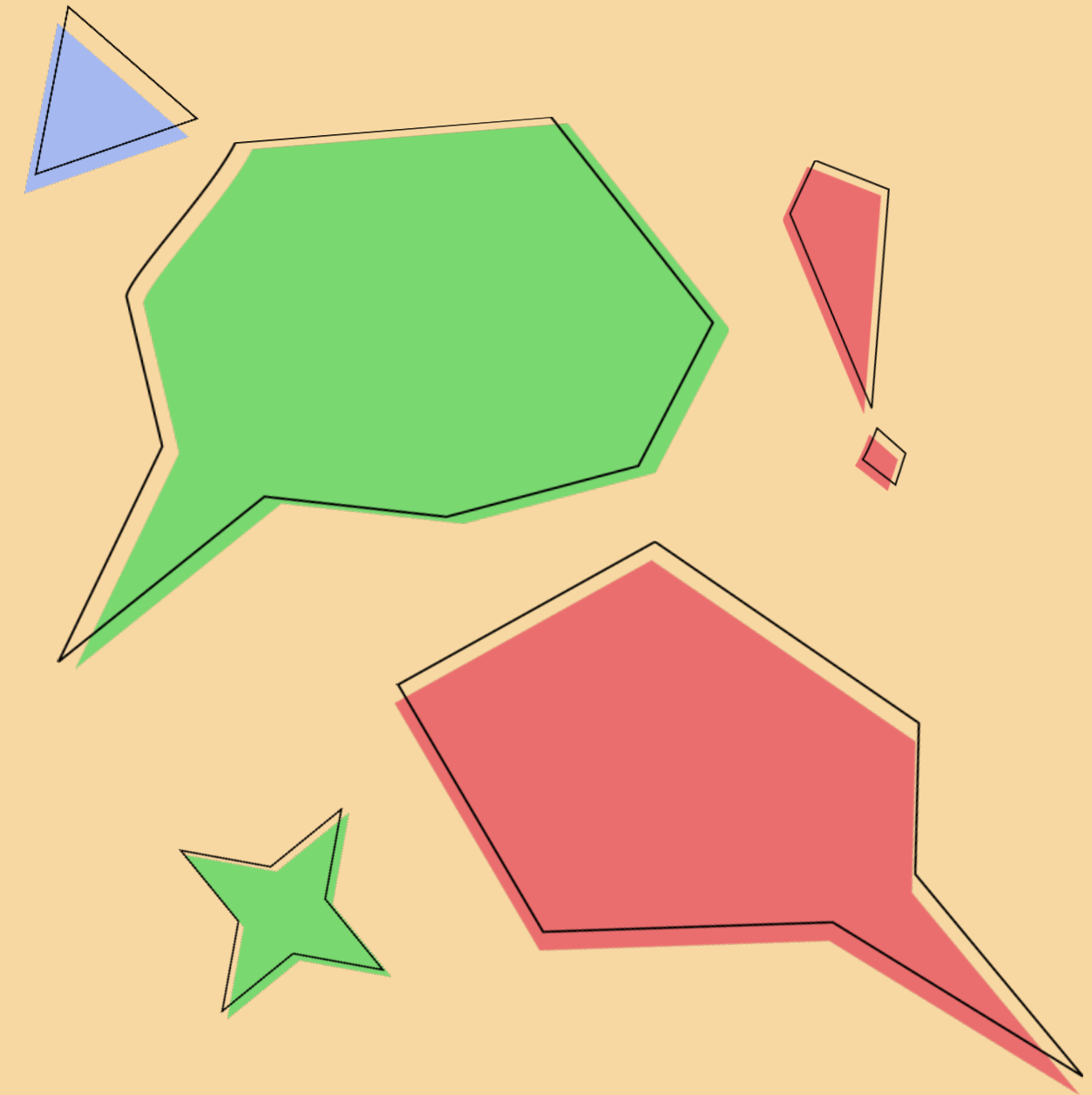


Обработка данных внутри структуры

Поля доступны по именам: например, можно прочитать только «класс» или обновить значение «статус» без вмешательства в остальные признаки. Это делает операции точными и снижает вероятность перепутать параметры.

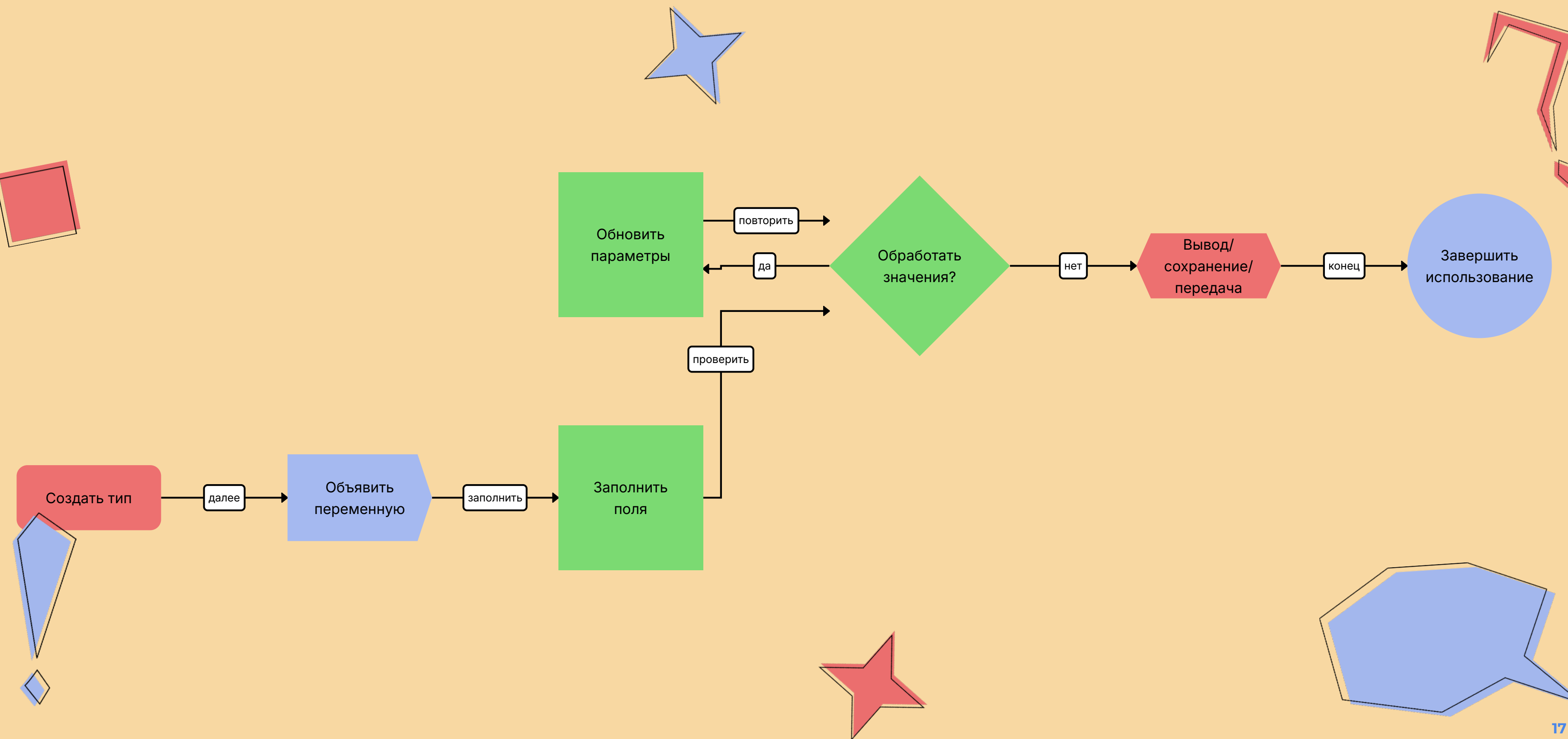
Отдельные значения меняются независимо, а сама запись передается как единая сущность: в процедуры и функции. Так удобнее строить алгоритмы сортировки, поиска и вывода, работая с целым объектом сразу.

Частые действия включают сравнение объектов по ключевым полям, выбор нужных записей и форматированный вывод всей структуры. Такой подход хорошо подходит для списков однотипных элементов и дальнейшей обработки результатов.



Жизненный цикл объекта с составными данными

От описания типа до завершения использования объекта в программе



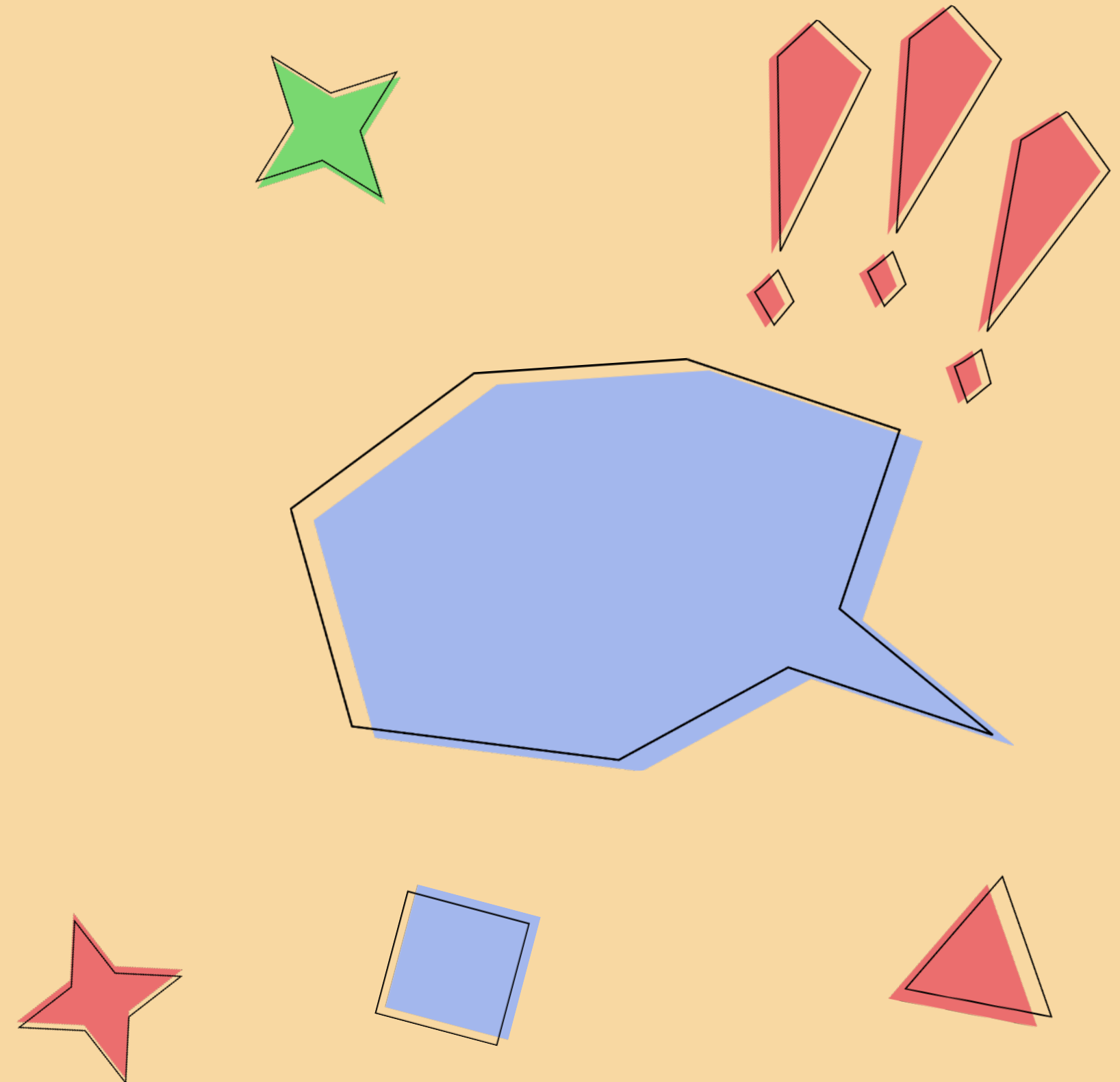
Типичные ошибки при работе со структурами

Частая путаница — смешивание типа данных и конкретной переменной этого типа. В результате программист может описывать поля неверно или обращаться к данным так, будто структура уже заполнена, что вызывает сбои.

Другая ошибка — забыть заполнить часть полей. Тогда алгоритм получает неполные сведения, а вывод и сравнения работают с некорректными значениями, что приводит к логическим ошибкам даже при успешной компиляции.

Нередко смешивают сведения разных объектов в одной записи. Например, в «заказ» попадают параметры из другой карточки, и дальше поиск по номеру или подсчёт суммы дадут неверные результаты из-за разорванной смысловой связи.

Также возникают проблемы при обращении к полю с неверным именем или при несоответствии типов значений. Это приводит либо к ошибкам компиляции, либо к неправильной обработке данных, когда программа «понимает» поле иначе.

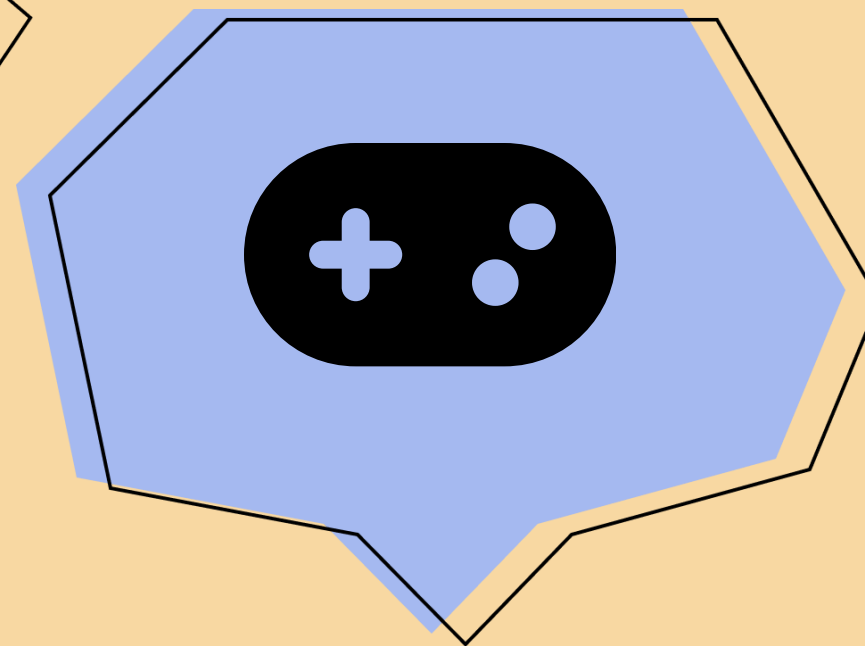
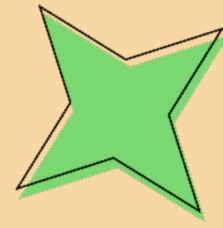
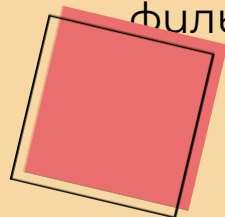


Практическая значимость комбинированных типов



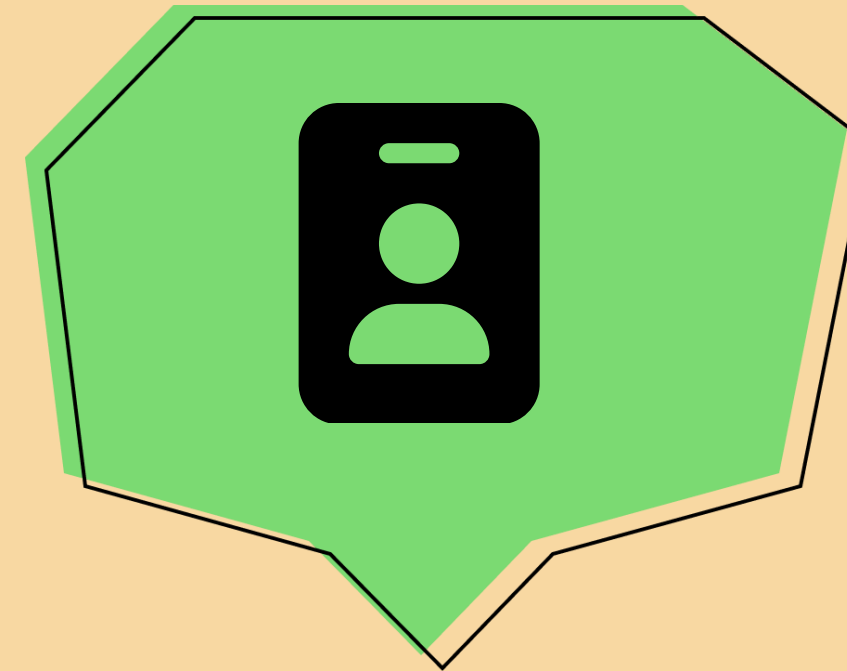
Базы данных и информационные системы

Структуры позволяют хранить данные как «карточки»: пользователь, товар, документ или запись. Благодаря группировке полей проще поддерживать целостность информации и выполнять операции поиска, фильтрации и обновления.



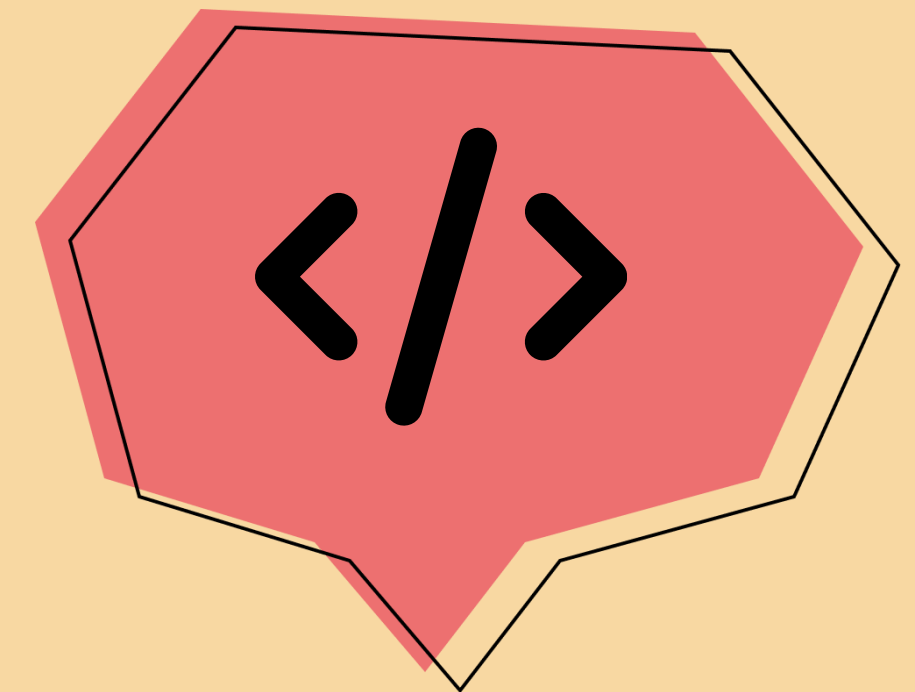
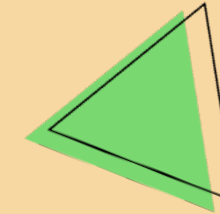
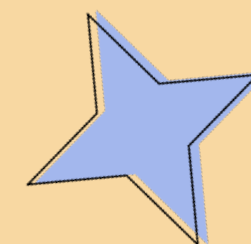
Игровые программы и симуляции

Игровые объекты требуют набора признаков: характеристики персонажа, состояние, параметры инвентаря. Комбинированные типы помогают описывать эти объекты целиком и обрабатывать их в циклах обновления без путаницы параметров.



Интерфейсы и формы ввода

В формах часто нужно связать имя, параметры и статусы в единую сущность, чтобы корректно проверять ввод и собирать результат. Структуры упрощают передачу данных от интерфейса к обработчикам и обратно.



Переход от модели задачи к алгоритму

Комбинированные типы помогают формализовать описание с естественного языка: «ученик с оценками», «заказ с суммой». Так объект становится удобным аргументом для алгоритмов, и код ближе к смыслу задачи.



Итоги: комбинированный тип данных — основа точного моделирования

Комбинированный тип объединяет связанные поля в одну логическую сущность, делая модель объекта точнее. Он повышает структурированность программы, облегчает хранение и обработку сложных данных и снижает число ошибок. Эффект зависит от правильного выбора полей и глубины вложенности.

