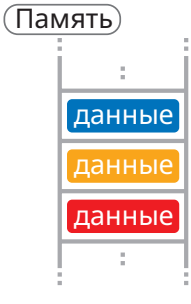
No.
1-1 Что такое структура данных

Порядок и расположение данных

В компьютере данные хранятся в памяти. Память можно представить в виде ряда ячеек, как показано на рисунке ниже. В каждой из этих ячеек хранится один фрагмент данных.



При хранении данных в памяти структура данных определяет их порядок и расположение.

Структура данных телефонного справочника

► Пример 1: естественный порядок

В качестве примера возьмем записную книжку с номерами телефонов. В наши дни обычно люди пользуются мобильными телефонами и хранят телефонные номера и другие контактные данные непосредственно в памяти своего устройства, но мы мысленно перенесемся в те времена, когда все приходилось записывать на бумаге. Пусть каждый раз, желая сохранить чей-то номер телефона, вы пишете имя человека и номер на странице записной книжки.

Имя	Номер телефона
Миясита Такеши	090-uuu-uuuu
Кавада Мика	090-xxxx-xxxx
Нозаки Дзюнко	090-yyy-yyyy
Магазин сакэ «Асада»	03-zzz-zzzz
…	…

Предположим, вы хотите позвонить госпоже Йоко Ямасита. Имена и номера телефонов просто записаны подряд, одно за другим. Скорее всего, вы не помните, где именно записали номер телефона госпожи Ямасита. Единственный способ узнать это — просмотреть все номера по порядку, начиная с первого. Вы можете начать просмотр с конца или вообще проверять записи наугад, но по затратам времени и усилий это не будет сильно отличаться, если просто просмотреть все записи, начиная с первой. Если в вашей записной книжке всего несколько номеров, вы с первого взгляда найдете нужный. Но если, допустим, номеров около 500, то, согласитесь, вам будет непросто.

► Пример 2: алфавитный порядок

Теперь давайте упорядочим записи в алфавитном порядке по именам. Поскольку они расположены в определенном порядке, эти данные уже имеют некоторую структуру:

Имя	Номер телефона
Акияма Юко	090-aaa-aaaa
Анда Юя	090-bbb-bbbb
«Асада», магазин сакэ	03-zzz-zzzz
Акаи Наоки	090-ccc-cccc
…	…

Это облегчает поиск нужного человека. По первой букве имени можно определить примерное местоположение номера. Но как быть, если нужно добавить новую запись, сохранив алфавитный порядок? Недавно у вас появился знакомый по имени Акутсу Хироси, который дал вам свой номер телефона. Если придерживаться алфавитного порядка, то господин Акутсу должен быть помещен между господином Акиямой и госпожой Анда. Однако, как видно из таблицы, свободного места нет, поэтому нам придется переместить телефон госпожи Анда вниз по очереди.

Для этого необходимо выполнить целую последовательность действий: добавить пустую строку в конце, переписать содержимое всех строк, начиная с конца, а потом записать новое имя и телефон. При наличии 500 элементов данных нам не хватит и часа, даже если одна операция займет всего 10 секунд.

► Плюсы и минусы

Подводя итог вышесказанному, повторим, что метод расположения данных в порядке их поступления прост, поскольку при вставке новых данных нужно сделать лишь одну запись, но на поиск по такому списку всегда будет уходить много времени. С другой стороны, метод расположения данных в алфавитном порядке удобен для поиска, но потребует больше времени и усилий для вставки новых записей.

Оба способа имеют свои преимущества и недостатки, выбор метода зависит от того, как вы будете использовать телефонную книгу. Если вы вряд ли будете обновлять телефонную книгу после ее создания, то лучше выбрать второй способ. С другой стороны, если вы часто вставляете данные, но ищете записи редко, вам следует использовать первый способ.

► А что, если объединить естественный и алфавитный порядок?

Покажем, как можно объединить оба способа. Достаточно взять отдельную таблицу для каждой строки хираганы[1], например «А-ряд», «К-ряд», «С-ряд» и т. д. В рамках одной таблицы будем придерживаться естественного порядка и добавлять записи в порядке поступления.

А-ряд

Имя	Номер телефона
Акаи Наоки	090-aaa-aaaa
Акияма Юко	03-zzz-zzzz
Анда Юя	090-zzz-zzzz
«Асада», магазин сакэ	090-aaa-aaaa
…	…

[1] Японская слоговая азбука, одна из составляющих японской письменности. — *Примеч. ред.*

К-ряд

Имя	Номер телефона
Киношита Хироси	090-aaa-aaaa
Кодзи Кагава	090-bbb-bbbb
Кумано Дзюнко	03-zzz-zzzz
Курумаки Тору	090-ccc-cccc
…	…

С-ряд

Имя	Номер телефона
Сенкава Гаку	03-aaa-aaaa
…	…
…	…
…	…
…	…

Таким образом, при добавлении новых данных вы можете просто записать их в конец соответствующей таблицы. Конечно, искать придется с самого начала, но это лучше, чем искать по всей телефонной книге.

Концепция структуры данных аналогична. Когда данные хранятся в памяти, они должны быть структурированы. Структуры данных строятся в соответствии с решаемыми задачами хранения и обработки данных в памяти и позволяют повысить эффективность их использования.

Что вы узнаете из главы 1?

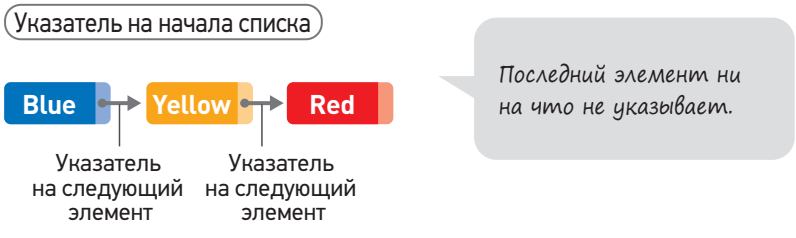
В этой главе рассматриваются семь структур данных. Как уже говорилось выше, данные располагаются в памяти последовательно, линейно. Но мы покажем, как можно создавать более сложные, нелинейные, структуры данных, с помощью специального инструмента — указателей. В остальных разделах книги вы найдете немало примеров применения всевозможных структур данных в различных задачах. Например, о применении графов рассказывается в разделе 4.2 «Поиск в ширину».

См. также: 4.2. «Поиск в ширину», с. 096.

No.
1-2 Список

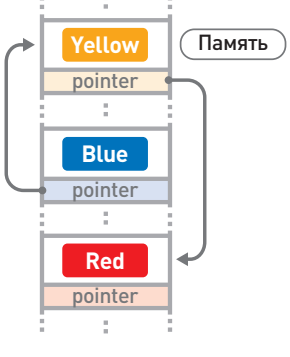
Список — это структура данных, в которой они расположены последовательно. Несмотря на простоту добавления и удаления данных, доступ к ним осуществляется медленно.

01



Это схематичное изображение списка. Здесь данные — это три блока разного цвета: синий, желтый и красный, обозначенные Blue, Yellow и Red соответственно. Каждый фрагмент данных также содержит поле указателя (pointer), который указывает на следующий фрагмент данных в памяти.

02



В списках данные не обязательно должны храниться в смежных областях памяти.

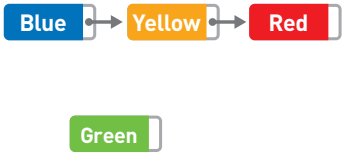
03

Поскольку данные хранятся по частям, доступ к каждой части можно получить только путем обхода всех указателей с самого начала (это называется *последовательным*, или *секвенциальным*, доступом). Например, если нужно получить доступ к красному блоку Red, сначала мы получим доступ к самому первому, синему блоку Blue.

04

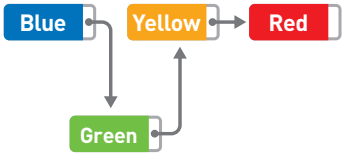
Чтобы добраться до блока Red, нам нужно перейти к нему через блок Yellow.

05



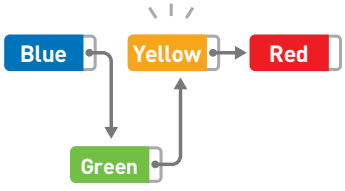
Добавлять данные в список очень просто: достаточно изменить указатель до и после добавляемых данных. Например, можем добавить новый, зеленый блок Green между блоками Blue и Yellow.

06



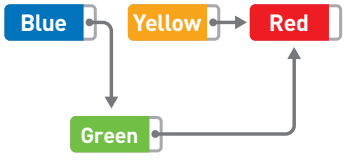
Добавление данных завершается перестановкой указателей с Blue на Green и с Green на Yellow.

07



Данные можно удалить таким же образом, изменяя поле указателя. Например, допустим, что нам нужно удалить блок Yellow.

08



В этом случае мы просто изменим поле указателя зеленого блока с Yellow на Red. Блок Yellow остается в памяти, но к нему нельзя обратиться. Когда в будущем эта область будет использоваться снова, она будет перезаписана и использована повторно.

ОЦЕНКА ВРЕМЕНИ ВЫЧИСЛЕНИЙ

Как обстоит дело со временем вычисления при использовании списков? Пусть у нас имеются данные, хранящиеся в списке. Пусть *n* — количество данных. Доступ к данным мы начинаем с начала списка (это называется *«линейный поиск»*), поэтому если интересующие нас данные находятся в конце списка, это займет O(*n*) времени. С другой стороны, данные могут быть добавлены за постоянное время O(1), которое не зависит от n, так как при этом надо обновить только два указателя. Конечно, при этом предполагается, что позиция, которую вы хотите добавить, уже найдена и известна. Аналогичным образом за время O(1) может быть выполнено удаление.

См. также: 3.1. «Линейный поиск», с. 086.

ДОПОЛНЕНИЕ

Такой список является самым базовым и может иметь несколько полезных расширений.

У последнего фрагмента данных в линейном списке поле указателя остается пустым; если его установить на первый элемент списка, это позволит создать циклический список. В циклическом списке отсутствует понятие «первый» или «последний».

Циклический список

Кроме того, в обычном списке каждый фрагмент данных имеет один указатель, но существует возможность создать двунаправленный список, который имеет два указателя на предыдущий и на последующий элементы.

Это удобно, поскольку список можно просматривать как с начала, так и с конца. Недостатком двунаправленного списка является то, что количество указателей увеличивается вдвое, поэтому для хранения данных требуется больше места. Кроме того, при добавлении или удалении данных приходится переназначать вдвое больше указателей.

Двунаправленный список