Уральский федеральный университет

СТРУКТУРЫ ДАННЫХ

Лектор
Спиричева Наталия Рахматулловна

Ст. преподаватель каф. ИТ
Ауд. Р-246

Структуры данных

Составитель курса лекций:
Спиричева Наталия Рахматулловна,
ст. преподаватель каф. Информационных технологий

Структуры данных и алгоритмы

Целью лекции является приобретение студентами следующих компетенций:

- знать свойства полустатических структур и их отличие от статических и динамических;
- иметь представление о физической и логической структуре;
- правильно и оптимально использовать полустатические структуры при программировании на ЯВУ.

Структуры данных и алгоритмы

Основные темы лекции:

- Характерные особенности полустатических структур
- Строки

ПОЛУСТАТИЧЕСКИЕ СТРУКТУРЫ ДАННЫХ

Характерные особенности полустатических структур

Характерные особенности полустатических структур

- Полустатические структуры данных характеризуются такими признаками:
- имеют переменную длину и простые процедуры ее изменения;
- изменение длины структуры происходит в определенных пределах, не превышая какого-то максимального (предельного) значения.

Характерные особенности полустатических структур

Если полустатическую структуру рассматривать на логическом уровне, то о ней можно сказать, что это последовательность данных, связанная отношениями линейного списка. Доступ к элементу может осуществляться по его порядковому номеру.

Физическое представление полустатических структур данных в памяти - это обычно последовательность слотов в памяти, где каждый следующий элемент расположен в памяти в следующем слоте (т.е. вектор). Физическое представление может иметь также вид однонаправленного связного списка (цепочки), где каждый следующий элемент адресуется указателем, находящимся в текущем элементе. В последнем случае ограничения на длину структуры гораздо менее строгие.

Характерные особенности полустатических структур

К полустатическим структурам относят:

- Строки
- Стеки
- Очереди
- Деки

Характерные особенности полустатических структур

Строки

Строки

Логическая структура

Строка - это линейно упорядоченная последовательность символов, принадлежащих конечному множеству символов, называемому алфавитом. Строки обладают следующими важными свойствами:

- их длина переменна, хотя алфавит фиксирован;
- обращение к символам строки идет с какого-нибудь одного конца последовательности, т.е важна упорядоченность этой последовательности, а не ее индексация; в связи с этим свойством строки часто называют также цепочками;
- целью доступа к строке является не отдельный ее элемент (хотя это тоже не исключается), а некоторая цепочка символов в строке - подстрока

Строки

В зависимости от ориентации языка программирования средства работы со строками занимают в языке более или менее значительное место. Рассмотрим три примера возможностей работы со строками.

Язык C является языком системного программирования, типы данных, с которыми работает язык C, максимально приближены к тем типам, с которыми работают машинные команды. Поскольку машинные команды не работают со строками, нет такого типа данных и в языке C. Строки в C представляются в виде массивов символов. Операции над строками могут быть выполнены как операции обработки массивов или же при помощи библиотечных (но не встроенных!) функций строковой обработки.

Строки

В языках универсального назначения обычно строковый тип является базовым в языке: STRING в PASCAL, CHAR(n) в PL/1. (В PASCAL длина строки, объявленной таким образом, может меняться от 0 до *n*, в PL/1, чтобы длина строки могла меняться, она должна быть объявлена с описателем VARING.) Основные операции над строками реализованы как простые операции или встроенные функции. Возможны также библиотеки, обеспечивающие расширенный набор строковых операций.

Язык REXX ориентирован прежде всего на обработку текстовой информации. Поэтому в REXX нет средств описания типов данных: все данные представляются в виде символьных строк. Операции над данными, не свойственные символьным строкам, либо выполняются специальными функциями, либо приводят к прозрачному для программиста преобразованию типов. Так, например, интерпретатор REXX, встретив оператор, содержащий арифметическое выражение, сам переводит его операнды в числовой тип, вычисляет выражение и преобразует результат в символьную строку. Целый ряд строковых операций является простыми операциями языка, а встроенных функций обработки строк в REXX несколько десятков.

Строки

Операции над строками

Базовыми операциями над строками являются:

- определение длины строки;
- присваивание строк;
- конкатенация (сцепление) строк;
- выделение подстроки;
- поиск вхождения.

Операция определения длины строки имеет вид функции, возвращаемое значение которой - целое число - текущее число символов в строке.
Операция присваивания имеет тот же смысл, что и для других типов данных.

Строки

Операция сравнения строк имеет тот же смысл, что и для других типов данных. Сравнение строк производится по следующим правилам: сравниваются первые символы двух строк. Если символы не равны, то строка, содержащая символ, место которого в алфавите ближе к началу, считается меньшей. Если символы равны, сравниваются вторые, третьи и т.д. символы. При достижении конца одной из строк, строка меньшей длины считается меньшей. При равенстве длин строк и попарном равенстве всех символов в них строки считаются равными.

Строки

Результатом операции сцепления двух строк является строка, длина которой равна суммарной длине строк-операндов, а значение соответствует значению первого операнда, за которым непосредственно следует значение второго операнда. Операция сцепления дает результат, длина которого в общем случае больше длин операндов. Как и во всех операциях над строками, которые могут увеличивать длину строки (присваивание, сцепление, сложные операции), возможен случай, когда длина результата окажется большей, чем отведенный для него объем памяти. Естественно, эта проблема возникает только в тех языках, где длина строки ограничивается. Возможны три варианта решения этой проблемы, определяемые правилами языка или режимами компиляции:

- никак не контролировать такое превышение; возникновение такой ситуации неминуемо приводит к трудно локализуемой ошибке при выполнении программы;
- завершать программу аварийно с локализацией и диагностикой ошибки;
- ограничивать длину результата в соответствии с объемом отведенной памяти.

Строки

- Операция выделения подстроки выделяет из исходной строки последовательность символов, начиная с заданной позиции *n* с заданной длиной *l*.
- При реализации операции выделения подстроки в языке программирования и в пользовательской процедуре обязательно должно быть определено правило получения результата для случая, когда начальная позиция *n* задана такой, что оставшаяся за ней часть исходной строки имеет длину, меньшую заданной длины *l*, или даже *n* превышает длину исходной строки. Возможные варианты такого правила:
- аварийное завершение программы с диагностикой ошибки;
- формирование результата меньшей длины, чем задано, возможно даже - пустой строки.

Строки

Операция поиска вхождения находит место первого вхождения подстроки-эталона в исходную строку. Результатом операции может быть номер позиции в исходной строке, с которой начинается вхождение эталона, или указатель на начало вхождения. В случае отсутствия вхождения результатом операции должно быть некоторое специальное значение, например, нулевой номер позиции или пустой указатель.

Строки

На основе базовых операций могут быть реализованы и любые другие, даже сложные операции над строками. Например, операция удаления из строки символов с номерами от *n*1 до *n*2, включительно, может быть реализована как последовательность следующих шагов:

- ✔ выделение из исходной строки подстроки, начиная с позиции *1*, длиной (*n*1-1) символов;
- ✔ выделение из исходной строки подстроки, начиная с позиции (*n*2+1), длиной, равной длине исходной строки минус *n*2;
- ✔ сцепление подстрок, полученных на предыдущих шагах.

Строки

Представление строк в памяти

Представление строк в памяти зависит от того, насколько изменчивыми являются строки в каждой конкретной задаче, и средства такого представления варьируются от абсолютно статического до динамического. Универсальные языки программирования в основном обеспечивают работу со строками переменной длины, но максимальная длина строки должна быть указана при ее создании. Если программиста не устраивают возможности или эффективность тех средств работы со строками, которые предоставляет ему язык программирования, то он может либо определить свой тип данных "строка" и использовать для его представления средства динамической работы с памятью, либо сменить язык программирования на специально ориентированный на обработку текста (CNOBOL, REXX), в которых представление строк базируется на динамическом управлении памятью.

Строки

ВЕКТОРНОЕ ПРЕДСТАВЛЕНИЕ СТРОК.

Представление строк в виде векторов, принятое в большинстве универсальных языков программирования, позволяет работать со строками, размещенными в статической памяти. Кроме того, векторное представление позволяет легко обращаться к отдельным символам строки как к элементам вектора - по индексу.

Самым простым способом является представление строки в виде вектора постоянной длины. При этом в памяти отводится фиксированное количество байт, в которые записываются символы строки и нуль-символ (\0). Нуль-символ – символ с кодом равным 0. По ниму определяется длина строки. Если строка меньше отводимого под нее вектора, то после символов ставится нуль-символ, а лишние места хранят остаточную информацию памяти, которая нас не интересует, а если строка выходит за пределы вектора, то лишние (обычно справа строки) символы должны быть отброшены.

Строки

ПРЕДСТАВЛЕНИЕ СТРОК ВЕКТОРОМ ПЕРЕМЕННОЙ ДЛИНЫ С ПРИЗНАКОМ КОНЦА.

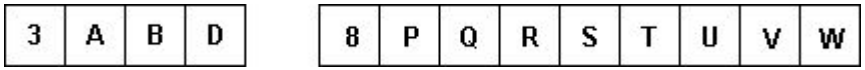
Признак конца - это особый символ, принадлежащий алфавиту (таким образом полезный алфавит оказывается меньше на один символ), и занимает то же количество разрядов, что и все остальные символы. Издержки памяти при этом способе составляют 1 символ на строку. Специальный символ-маркер конца строки в языке С называется нуль символом, и обозначается ‘\0’.

A	B	C	D	\0

P	Q	R	S	T	U	V	W	\0

ПРЕДСТАВЛЕНИЕ СТРОК ВЕКТОРОМ ПЕРЕМЕННОЙ ДЛИНЫ СО СЧЕТЧИКОМ

Счетчик символов - это целое число, и для него отводится достаточное количество битов, чтобы их с избытком хватало для представления длины самой длинной строки, какую только можно представить в данной машине. Обычно для счетчика отводят от 8 до 16 бит. Тогда при таком представлении издержки памяти в расчете на одну строку составляют 1-2 символа. При использовании счетчика символов возможен произвольный доступ к символам в пределах строки, поскольку можно легко проверить, что обращение не выходит за пределы строки. Счетчик размещается в таком месте, где он может быть легко доступен - в начале строки или в дескрипторе строки. Максимально возможная длина строки, таким образом, ограничена разрядностью счетчика.



Строки

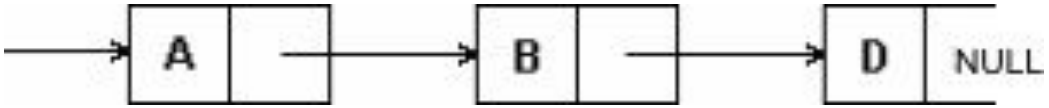
ВЕКТОР С УПРАВЛЯЕМОЙ ДЛИНОЙ.

Память под вектор с управляемой длиной отводится при создании строки, и ее размер и размещение остаются неизменными все время существования строки. В дескрипторе такого вектора-строки может отсутствовать начальный индекс, так как он может быть зафиксирован раз и навсегда установленными соглашениями, но появляется поле текущей длины строки.

Размер строки, таким образом, может изменяться от 0 до значения максимального индекса вектора. "Лишняя" часть отводимой памяти может быть заполнена любыми кодами - она непринимается во внимание при оперировании со строкой. Поле конечного индекса может быть использовано для контроля превышения строкой объема отведенной памяти.

ОДНОНАПРАВЛЕННЫЙ ЛИНЕЙНЫЙ СПИСОК

Каждый символ строки представляется в виде элемента связного списка; элемент содержит код символа и указатель на следующий элемент. Одностороннее сцепление предоставляет доступ только в одном направлении вдоль строки. На каждый символ строки необходим один указатель, который обычно занимает 4 байта.

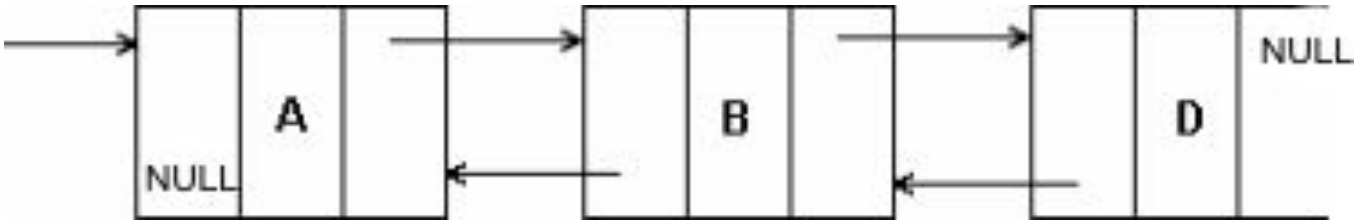


Строки

ДВУНАПРАВЛЕННЫЙ ЛИНЕЙНЫЙ СПИСОК

В каждый элемент списка добавляется также указатель на предыдущий элемент.

Двустороннее сцепление допускает двустороннее движение вдоль списка, что может значительно повысить эффективность выполнения некоторых строковых операций. При этом на каждый символ строки необходимо два указателя , т.е. 8 байт.



Строки

БЛОЧНО-СВЯЗНОЕ ПРЕДСТАВЛЕНИЕ СТРОК

Такое представление позволяет в большинстве операций избежать затрат, связанных с управлением динамической памятью, но в то же время обеспечивает достаточно эффективное использование памяти при работе со строками переменной длины.

Строки

МНОГОСИМВОЛЬНЫЕ ЗВЕНЬЯ ФИКСИРОВАННОЙ ДЛИНЫ

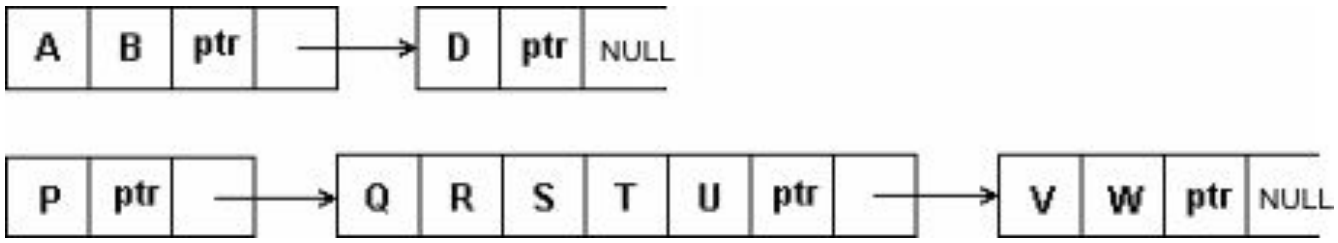
Многосимвольные группы (звенья) организуются в список так, что каждый элемент списка, кроме последнего, содержит группу элементов строки и указатель следующего элемента списка. Поле указателя последнего элемента списка хранит признак конца - пустой указатель. В процессе обработки строки из любой ее позиции могут быть исключены или в любом месте вставлены элементы, в результате чего звенья могут содержать меньшее число элементов, чем было первоначально. По этой причине необходим специальный символ, который означал бы отсутствие элемента в соответствующей позиции строки. Обозначим такой символ 'emp'[illegible] символов, из которых орга[illegible]

Строки

МНОГОСИМВОЛЬНЫЕ ЗВЕНЬЯ ПЕРЕМЕННОЙ ДЛИНЫ

Переменная длина блока дает возможность избавиться от пустых символов и тем самым экономить память для строки. Однако появляется потребность в специальном символе - признаке указателя 'ptr'.

С увеличением длины групп символов, хранящихся в блоках, эффективность использования памяти повышается. Однако негативной характеристикой рассматриваемого метода является усложнение операций по резервированию памяти для элементов списка и возврату освободившихся элементов в общий список доступной памяти.

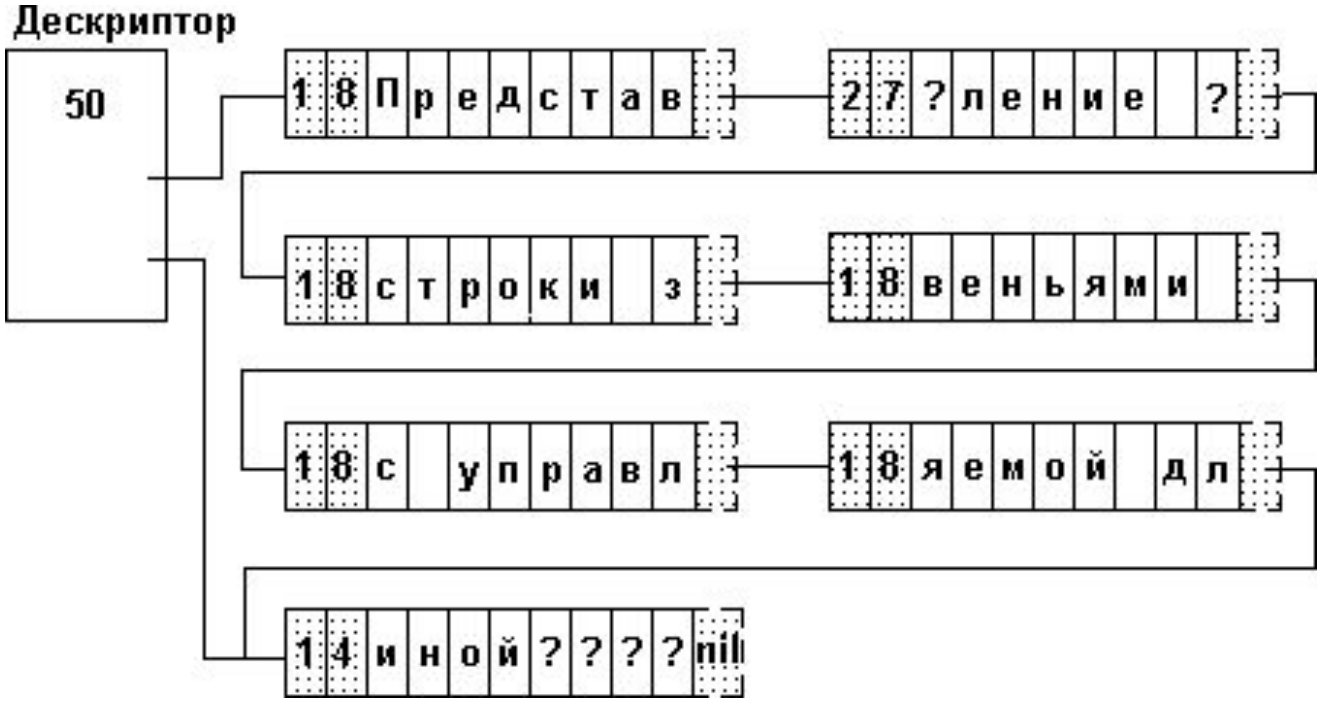


Строки

МНОГОСИМВОЛЬНЫЕ ЗВЕНЬЯ С УПРАВЛЯЕМОЙ ДЛИНОЙ

Память выделяется блоками фиксированной длины. В каждом блоке помимо символов строки и указателя на следующий блок содержатся номера первого и последнего символов в блоке. При обработке строки в каждом блоке обрабатываются только символы, расположенные между этими номерами. Признак пустого символа не используется: при удалении символа из строки оставшиеся в блоке символы уплотняются и корректируются граничные номера. Вставка символа может быть выполнена за счет имеющегося в блоке свободного места, а при отсутствии такового - выделением нового блока. Хотя операции вставки/удаления требуют пересылки символов, диапазон пересылок ограничивается одним блоком. При каждой операции изменения может быть проанализирована заполненность соседних блоков и два полупустых соседних блока могут быть переформированы в один блок. Для определения конца строки может использоваться как пустой указатель в последнем блоке, так и указатель на последний блок в дескрипторе строки. Последнее может быть весьма полезным при выполнении некоторых

Строки



Представление строки звеньями управляемой длины

Контрольные вопросы

1.Каковы особенности полустатических структур данных?

2. Перечислите основные полустатические структуры?

3. Какие основные операции выполняются над полустатическими структурами?

Спасибо за внимание!