


Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

Учебный курс

Методы программирования - 2

Нижегородский государственный университет им. Н.И. Лобачевского
Институт информационных технологий, математики и механики

Учебный курс:

Методы программирования - 2

Тема 5:

Структуры хранения с использованием указателей (списки)

Гергель В.П., профессор ,
директор института ИТММ

Содержание

Глава 1. *Структура действия и структуры данных*

1.6. Структуры хранения с использованием указателей (*списки*)

1. Представление отношений следования с использованием указателей. Понятие линейного списка.
2. Реализация списков на языке высокого уровня.
   Практическая работа 6: Реализация стека
3. Реализация списков с использованием динамически распределяемой памяти. Пример разработки структуры хранения стека.
4. Примеры использования стеков: поразрядная сортировка, преобразование арифметических выражений в польскую форму записи
5. **Практическая работа 7**: Разработка общего представления линейного списка для обеспечения списковой структуры хранения
6. Стандартная библиотека шаблонов алгоритмического языка C++

Вопросы для обсуждения

1.6. Структуры хранения с использованием указателей…

1. Представление отношений следования с использованием указателей. Понятие линейного списка…

- ☑ Необходимость перепаковки для обеспечения динамического распределения памяти возникает в силу принятого способа реализации отношения следования - следующий элемент структуры располагается в следующем элементе памяти (с адресом, большим на 1)
- ☑ Устранение перепаковки возможно только при кардинальном изменении способа реализации основных отношений – необходимо допустить размещение следующих элементов структуры в произвольных элементах памяти (там, где имеется свободные области памяти)
- ⇘ Возможность такого подхода может быть обеспечена запоминанием для каждого текущего элемента структуры адреса памяти, где хранится следующий элемент
- ⇘ Интерпретация содержимого элемента памяти (значение или адрес следующего элемента) в самом простом варианте может быть обеспечена фиксированным форматом используемых участков памяти

1.6. Структуры хранения с использованием указателей…

1. Представление отношений следования с использованием указателей. Понятие линейного списка…

Определение 1.16. Под *квантом памяти* понимается последовательность элементов памяти с последовательно-возрастающими адресами. *Именем* (адресом) этой группы считается адрес первого слова кванта. Элементы кванта называются *полями*.

Определение 1.17. В общем случае, набор элементов памяти, связанных с одним именем, называют *звеном*.

☑ Далее будут использоваться двухэлементные звенья памяти, в которых первое поле будет использоваться для хранения значений, а второе поле – для запоминания адресов.

1.6. Структуры хранения с использованием указателей…

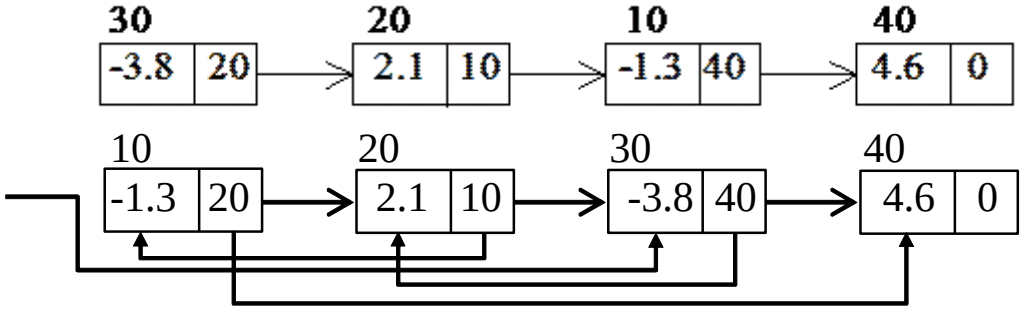
1. Представление отношений следования с использованием указателей. Понятие линейного списка…

Определение 1.18. Способ задания отношения следования, в котором фиксация месторасположения следующего элемента производится путем запоминания соответствующего адреса памяти, называется *сцеплением* (пары, хранящие a_i и a_{i+1}, сцеплены адресными указателями).

1.6. Структуры хранения с использованием указателей…

1. Представление отношений следования с использованием указателей. Понятие линейного списка...

☑ Для изображения структуры хранения с использованием сцепления звенья памяти рисуются в виде прямоугольников, а сцепление звеньев показывается в виде стрелок.



Логический порядок звеньев

Физический порядок звеньев

☑ Индикация последнего звена в списке обычно производится записью в поле адреса некоторого *барьера* – фиктивного (неадресного) значения (как правило 0 или -1).

☑ Для доступа к звеньям списка должен быть известен адрес первого звена списка. Указатель, в котором этот адрес запоминается, называется *переменной связи*.

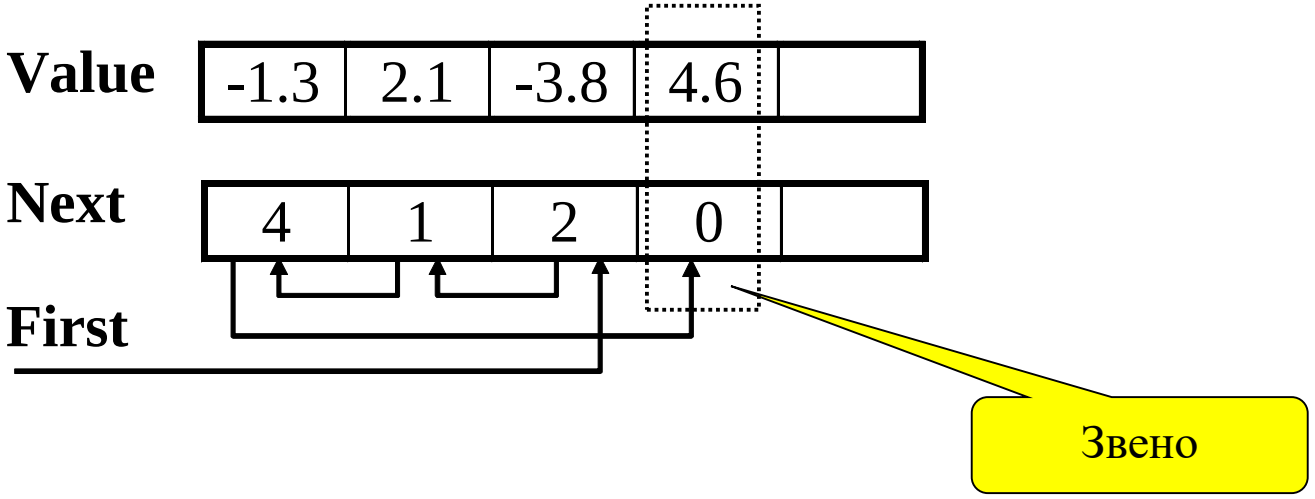
1.6. Структуры хранения с использованием указателей…

1. Представление отношений следования с использованием указателей. Понятие линейного списка.

Определение 1.19. Структура хранения данного типа (звенья, сцепление, барьер, переменная связи) называется *линейным* или *односвязным списком*.

2. Реализация списков на языке высокого уровня…

Подход 1. Для имитации звеньев могут быть использованы два массива, один из которых используется для хранения значений, другой – для хранения индексов следующих элементов. В этом случае, звено есть элементы массивов с одинаковым индексом, адрес (имя) звена – индекс массивов.



2. Реализация списков на языке высокого уровня…

Подход 2. С использованием ООП звено может быть представлено в виде объекта. Образ памяти, выделенной для хранения структур данных, в этом случае будет представлять массив звеньев-объектов.

```
class TLink {
  public:
    int  Value; // значение
    int  Next;  // индекс следующего звена
  protected:
    TLink();
};
TLink Mem[MemLimit];
```

1.6. Структуры хранения с использованием указателей…

2. Реализация списков на языке высокого уровня…

Практическая работа 6: Реализация стека

1.6. Структуры хранения с использованием указателей…

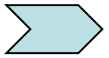
2. 2. Реализация списков на языке высокого уровня. Сравнение непрерывной и списковой структур хранения

	Непрерывная память	Списки
1	Перепаковка для динамического распределения памяти	Динамическое распределение памяти эффективно реализуется при помощи списка свободных звеньев
2	В структуре хранения хранятся только данные	В структуре хранения хранятся данные и указатели
3	К элементам структуры данных беспечивается прямой доступ	К элементам структуры данных обеспечивается последовательный доступ

1.6. Структуры хранения с использованием указателей…

3. Реализация списков с использованием динамически распределяемой памяти…

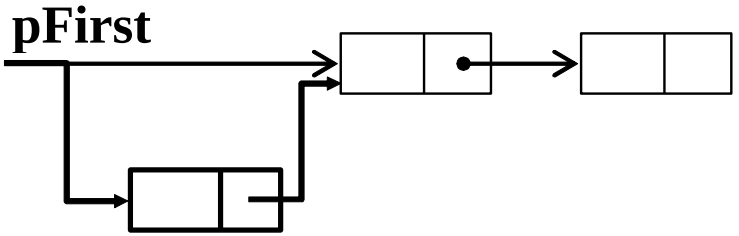
☑ Среда выполнения обеспечивает *динамически-распределяемую область памяти*:

– звено – 

– выделение звена – pTemp = new TDatLink()

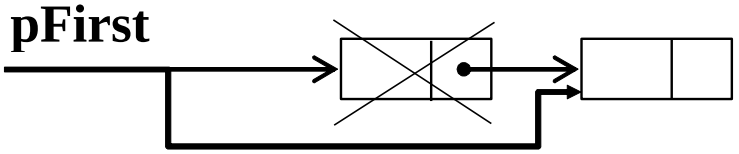
– освобождение звена – delete pTemp

Вставка в стек

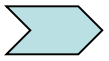



```
PTDatLink pTemp;
pTemp = new TDatLink();
pTemp->SetDatValue(Val);
pTemp->SetNextLink(pFirst);
pFirst = pTemp;
```

Выборка из стека



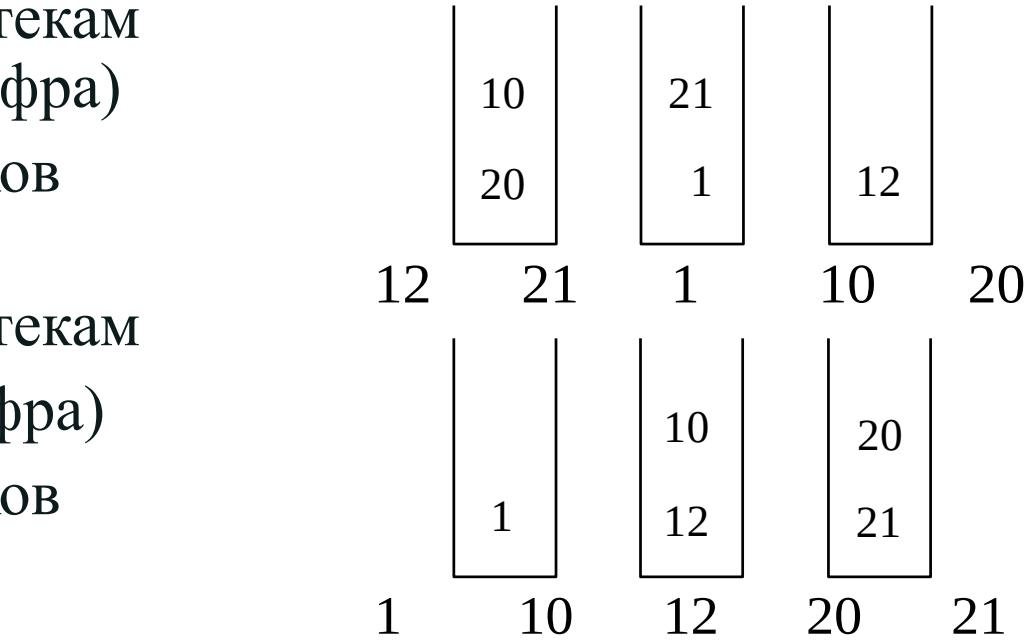

```
PTDatLink pTemp = pFirst;
Val = pFirst->GetDatValue();
pFirst = pFirst->GetNextLink();
delete pTemp;
```

Стек  Пример: программа, приложение

4. Пример использования стеков: поразрядная сортировка

Пример 1.9. Упорядочение данных методом поразрядной сортировки (на примере набора значений 20 12 1 21 10)

- Разложить значения по стекам (номер стека - младшая цифра)
- Собрать значения из стеков (от старшего стека)
- Разложить значения по стекам (номер стека - старшая цифра)
- Собрать значения из стеков (от младшего стека)



Пример: программа, приложение

4. Пример использования стеков: преобразование арифметических выражений в польскую форму записи …

Пример 1.10. Преобразование выражений из инфиксной формы в польскую запись

Формат записи выражения

- Выражение синтаксически правильно (без ошибок)
- Допускаются только однобуквенные идентификаторы для операндов
- В записи выражения нет пробелов; выражение заканчивается знаком '='

A+(B-C)*D-F/(G+H)=

4. Пример использования стеков: преобразование арифметических выражений в польскую форму записи …

Пример 1.10. Преобразование выражений из инфиксной формы в польскую запись

Алгоритм

1. Для операций вводится приоритет
   '*' '/' (3), '+' '-' (2), '(' (1), '=' (0)
2. Для хранения данных используется 2 стека (1 – для результата, 2 – для операций)
3. Исходное выражение просматривается слева направо
4. Операнды по мере их появления помещаются в стек 1
5. Символы операций и левые скобки помещаются в стек 2
6. При появлении правой скобки последовательно изымаются элементы из стека 2 и переносятся в стек 1. Данные действия продолжаются либо до опустошения стека 2 либо до попадания в стеке 2 на левую скобку
7. Если текущая операция, выделенная при обходе выражения, имеет меньший (более низкий) приоритет, чем операция на вершине стека 2, то такие операции из стека 2 переписываются в стек 1

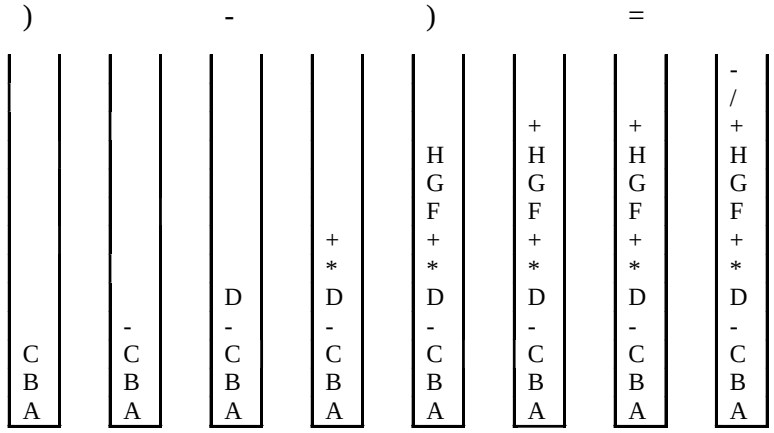
1.6. Структуры хранения с использованием указателей

4. Пример использования стеков: преобразование арифметических выражений в польскую форму записи …

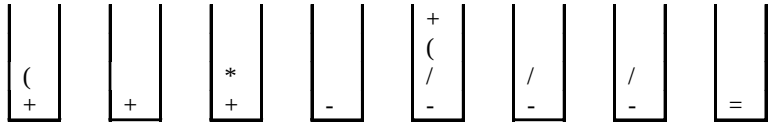
Пример 1.10. Преобразование выражений из инфиксной формы в польскую запись

Пусть выражение имеет вид A+(B-C)*D-F/(G+H)=

Стек 1 – выражение в постфиксной записи



Стек 2 - операции



программа,

приложение

1.6. Структуры хранения с использованием указателей…

5. Обеспечения списковой структуры хранения

Практическая работа 7: Разработка общего представления линейного списка

1.6. Структуры хранения с использованием указателей…

6. Стандартная библиотека шаблонов алгоритмического языка C++…

☑ В стандарте языка C++ предусматривается наличие в среде программирования *стандартной библиотеки шаблонов* (*Standard Template Library, STL*)

Основные понятия библиотеки STL

I. Библиотека включает в свой состав большое количество ***контейнеров***, представляющих собой структуры данных, в которых могут храниться объекты. В числе имеющихся контейнеров..

1.6. Структуры хранения с использованием указателей…

6. Стандартная библиотека шаблонов алгоритмического языка C++…

- vector<T> - вектор переменного размера,
- list<T> - двусвязный список,
- queue<T> - очередь,
- stack<T> - стек,
- deque<T> - дек,
- priority_queue<T> - приоритетная очередь,
- set<T> - множество,
- multiset<T> - множество с повторением элементов,
- map<key,val> - ассоциативный массив (таблица),
- multimap<key,val> - ассоциативный массив с повторением ключей

6. Стандартная библиотека шаблонов алгоритмического языка C++…

II. Для быстрого и эффективного построения вычислительных процедур, библиотека обеспечивает ***итераторы*** для всех видов контейнеров, которые представляют унифицированный механизм последовательного доступа к элементам контейнеров.

Общая схема:

- <класс-контейнер>::iterator Iter; - объявление итератора,
- Iter = <объект-контейнер>.begin(); - установка на первый элемент,
- Iter != <объект-контейнер>.end(); - проверка на завершение,
- ++Iter – переход к следующему элементу

В зависимости от типа контейнера, итератор может обеспечивать прямой доступ, быть одно- или двух- направленным, предназначенным только для чтения или записи и др.

6. Стандартная библиотека шаблонов алгоритмического языка C++…

III.Библиотека содержит для контейнеров большое количество реализованных ***обобщенных алгоритмов***. В числе таких алгоритмов:

- for_each() - вызвать функцию для каждого элемента,
- find() - найти первое вхождение элемента,
- find_if() - найти первое соответствие условию,
- count() - подсчитать число вхождений элемента,
- count_if() - подсчитать число соответствий условию,
- replace() - заменить элемент новым значением,
- copy() - скопировать элементы,
- unique_copy() - скопировать только различные элементы,
- sort() - отсортировать элементы,
- merge() - объединить отсортированные последовательности и др.

6. Стандартная библиотека шаблонов алгоритмического языка C++…

IV. Рассмотрим пример использования контейнера списка.

Пример: программа, приложение

Заключение

- Реализация отношения следования при помощи сцепления (адресных указателей)
- Линейный список как структура хранения
- Реализация списков на языках высокого уровня
- Реализация списков с использованием динамически-распределяемой области памяти
- Использование списковых структур хранения на примере реализации стеков
- Разработка общего представления линейного списка для обеспечения списковой структуры хранения
- Общая характеристика стандартной библиотеки шаблонов

Вопросы для обсуждения

- Сравнение непрерывной и списковых структур хранения
- Разработка стандартных программ для работы со списками

Темы заданий для самостоятельной работы

- Реализация очередей с использованием списков
- Реализация двунаправленных списков

Следующая тема

- Структуры данных и конструирование математических моделей

Контакты

Нижегородский государственный университет им. Н.И. Лобачевского (www.unn.ru)

Институт информационных технологий, математики и механики (www.itmm.unn.ru)

603950, Нижний Новгород, пр. Гагарина, 23,
р.т.: (831) 462-33-56,

Гергель Виктор Павлович

(http://www.software.unn.ru/?dir=*17*)

E-mail: gergel@unn.ru