**Практическая работа на Списки**

Тема “Списочные структуры данных и алгоритмы над ними”

Контрольные вопросы

1. Какие структуры данных используют в программах?
2. Приведите примеры использования структур данных в программах?
3. Почему мы их используем?
4. Для чего служат общие структуры данных и алгоритмы их обработки?
5. Как их использует программист?

**Теоретические сведения**

Список - это совокупность объектов, называемых элементами списка, в которой каждый объект содержит информацию о местоположении связанного с ним объекта

Если список располагается в оперативной памяти, то, как правило, информация для поиска следующего объекта — это адрес (указатель) в памяти для С/С++ или ссылка в Java. Если список хранится в файле на диске, то информация о следующем элементе может включать смещение элемента от начала файла, положение указателя записи/считывания файла, ключ записи и любую другую информацию, позволяющую однозначно отыскать следующий элемент списка.

**Структура данных: Связанный список**

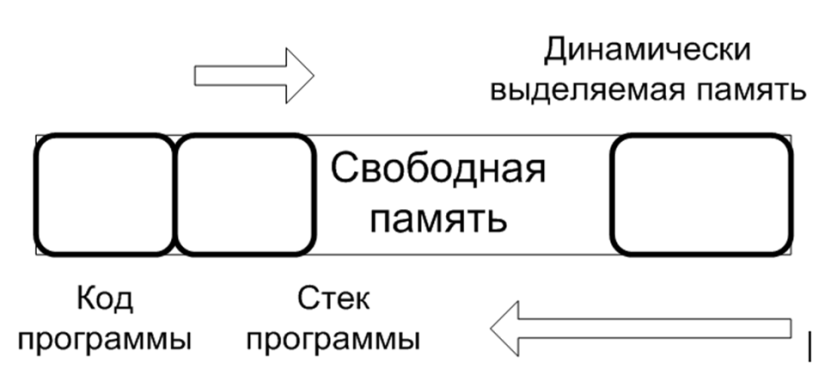
Действия над списками:

Create: создать элемент списка

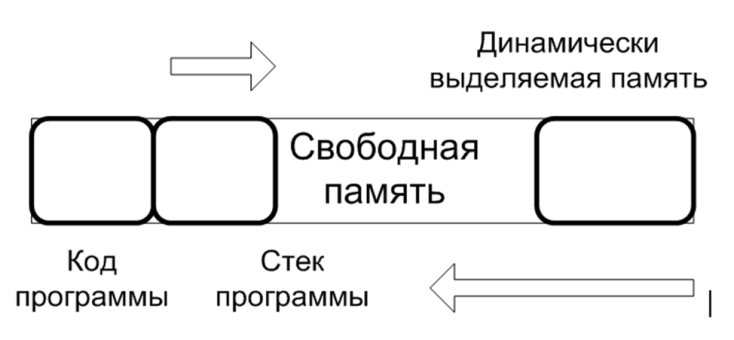
Add: добавить элемент списка

Search: найти элемент списка

Перед тем как мы начнем детальное рассмотрение динамических структур данных, вспомните, что вы знаете об архитектуре памяти компьютера. В общем случае для приложения доступны два вида памяти: стековая память, ее выделяется ограниченный размер и количество не может изменяться в ходе работы программы без специальных приемов. В данном виде памяти выделяется место для хранения локальных переменных, размещения параметров вызова функций и хранения точек возврата. Размер стека не слишком велик и для хранения больших массивов



Динамические структуры данных наиболее часто применяются для создания контейнеров – специальных типов данных для хранения определенного множества элементов, а также реализующих определенную дисциплину обслуживания (добавления, удаления, поиска, изменения)

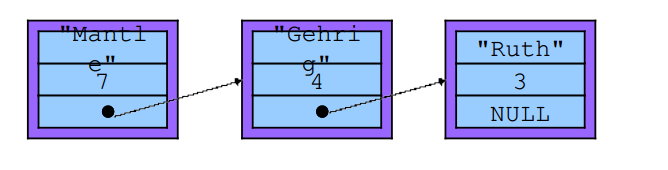


Наиболее простым типом контейнера является список. Поэтому мы и начали с него. Так, как количество элементов списка заранее не известно, то список определяется по следующему рекурсивному правилу: NULL – список (пустой список);

A = (x, B) – список, в данном случае “x” – элемент списка, “B” – список хранения (хранимых элементов) Рассмотрим пример объявления узла списка и заполнения его элементами 1..n.

|  |  |
| --- | --- |
| *public class Node {*  *private int data;*  *......*  *private Node next;*  *}* | Macintosh HD:Users:natalazorina:Desktop:Снимок экрана 2016-10-19 в 11.26.25.png |

Один или более элементов (узлов), которые хранят значения (информационное поле) и ссылку на следующий элемент (узел) списка



Действия над списками:

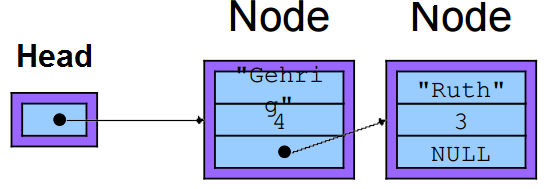
• Create: разместить структуру данных в памяти, начиная с 1 узла

• Add: вставить новы й узел с начала списка (at front of list)

• Search: организовать линейный поиск через весь список

• Remove: удалить неиспользуемый узел списка

Один или более элементов (узлов), которые хранят значения (информационное поле) и ссылку на следующий элемент (узел) списка



Алгоритмы:

• Create: разместить структуру данных в памяти, начиная с 1 узла

• Add: вставить новый узел с начала списка (at front of list)

• Search: организовать линейный поиск через весь список

• Remove: удалить неиспользуемый узел списка

**Пример реализации линейного списка без заглавного звена**

/\*\* \* Created by natalazorina on 03.04.15. \*/

**public class Node** {

Object data;

Node next;

public Node(Object dataValue){

next = null;

data = dataValue;

}

public Node(Object dataValue, Node nextValue){

next = nextValue;

data = dataValue;

}

public Object getData() { return data; }

public void setData(Object dataValue) {

data = dataValue;

}

 public Node getNext() { return next; }

public void setNext(Node nextValue) { next = nextValue; }

@Override

public String toString() {

String output ="";

output = "node data="+data;

return output;

}

}

**public class MyLinkedList** {

private Node head;

private int listCount;

public MyLinkedList() {

head = new Node(null);

listCount = 0;

}

public void add(Object data){

Node tmp = new Node(data);

Node current = head;

while (current.getNext() != null ){

current = current.getNext();

}

current.setNext(tmp);

listCount++;

}

public void add(Object data, int index){

Node tmp = new Node(data);

Node current = head;

for (int i=1; i<index && current.getNext() != null; i++){

current = current.getNext();

}

tmp.setNext(current.getNext());

current.setNext(tmp);

listCount++;

}

public Object get(int index){

if (index <= 0) return null;

Node current = head.getNext();

for (int i = 1; i < index; i++){

if (current.getNext() == null) return null;

current = current.getNext();

}

return current.getData();

}

public boolean remove(int index){

if (index < 1 && index > size()) return false;

Node current = head;

for (int i =1; i < index; i++){

if (current.getNext() == null) return false;

current = current.getNext();

}

current.setNext(current.getNext().getNext());

listCount--;

  return true;

}

public int size(){ return listCount; }

@Override public

String toString() {

Node current = head.getNext();

String output = "";

while (current != null){

output+="["+current.getData().toString()+"]";

current = current.getNext(); }

return output;

}

}

**Что нужно помнить при работе со писками:**

1. В языке Java выделение памяти выполняется с использованием new
2. Освобождение памяти происходит автоматически с помощью GC
3. Обозначение пустой ссылки null

**Задания по вариантам**

1. Создание двусвязного циклического списка вставками путем исключения первого элемента и включения в новый список с сохранением упорядоченности
2. Сортировка двусвязного списка путем исключения элемента с минимальным значением включением его в начало списка
3. Сортировка двусвязного циклического списка с перестановкой соседних элементов
4. Элемент двусвязного списка содержит ссылку на строку в динамической памяти. Написать методы просмотра списка и включения очередной строки с сохранением упорядоченности по длине строки и алфавиту
5. Элемент односвязного списка содержит массив из пяти целых переменных. Массив заполнен частично. Все значения хранятся в порядке возрастания. Написать метод включения значения в элемент списка с сохранением упорядоченности. При переполнении массива создается новый элемент списка
6. Элемент двусвязного циклического списка содержит ссылку на строку в динамической памяти. Написать методы м просмотра списка и включения очередной строки с сохранением упорядоченности по длине строки и алфавиту
7. Элемент двусвязного циклического списка содержит массив из пяти целых переменных. Массив может быть заполнен частично. Написать метод включения значения в элемент списка с сохранением упорядоченности. При переполнении массива создается новый элемент списка и в него включатся половина значений из переполненного
8. Элемент двусвязного списка содержит ссылку на строку. Строки упорядочены по возрастанию. Вставить строку в конец списка. В список помещается копия входной строки в динамической памяти
9. Элемент односвязного списка содержит ссылку на строку. Строки упорядочены по возрастанию. Вставить строку в список с сохранением упорядоченности. В список помещается копия входной строки в динамической памяти
10. Элемент односвязного списка содержит ссылку на строку. Отсортировать список путем исключения максимального элемента и включения его в начало списка
11. Сортировка односвязного списка рекурсивным методом. Метод разделяет список на две части относительно значения первого элемента и вызывает себя рекурсивно с полученными списками. Метод возвращает в качестве результата ссылку на отсортированный. Полученные от рекурсивного вызова списки “склеиваются” и возвращаются наверх.

Список литературы

1. «Структуры данных и алгоритмы» Автор: Альфред В. Ахо, Джон Э. Хопкрофт, Джеффри Д. Ульман Издательство: Вильямс Год: 2003 ISBN: 5-8459-0122-7
2. «Алгоритмы на Java. Фундаментальные алгоритмы и структуры данных», Роберт Седжвик, Кевин Уэйн, 4-е издание, 848 стр., ISBN 978-5-8459-1781-2, «ВИЛЬЯМС», 2013