Министерство образования Республики Беларусь
Учреждение образования «Полоцкий государственный университет»

Кафедра технологий программирования

Алгоритмы и структуры данных

Отчет по лабораторной работе №1

Вариант 11

Выполнил — Ланцев Евгений Николаевич. 21-ИТ-1, ФИТ

Проверил — преподаватель Виноградова А.Д.

Полоцк
2022 г.

Лабораторная работа № 1
“Реализация линейной структуры данных «Список» и основные алгоритмы обработки.”

Цель работы: ознакомиться с основами линейной структуры данных «Список», изучить основные алгоритмы обработки ЛСД «Список», научиться применять полученные знания на практике.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ (ответы на контрольные вопросы):

1. Определение понятия список.

В информатике, спи́сок (англ. list) — это абстрактный тип данных, представляющий собой упорядоченный набор значений, в котором некоторое значение может встречаться более одного раза.

2. Классификация списков по количеству полей указателей.

*По **количеству полей указателей** различают однонаправленный (односвязный) и двунаправленный (двусвязный) **списки***

3. Классификация списков по способу связи элементов.

*По **способу связи элементов** различают линейные и циклические **списки**.*

4. Виды списков. Отличие списков.
    1. Односвязный линейный список (ОЛС).
    2. Односвязный циклический список (ОЦС).
    3. Двусвязный линейный список
    4. Двусвязный циклический список

5. Основные действия, производимые над элементами односвязного линейного списка.
    - Инициализация списка
    - Добавление узла в список
    - Удаление узла из списка
    - Удаление корня списка
    - Вывод элементов списка
    - Взаимообмен двух узлов списка

6. Какие аргументы принимает функция добавления узла в список (ОЛС).

Функция добавления узла в список принимает два аргумента:

- Указатель на узел, после которого происходит добавление
- Данные для добавляемого узла.

7. Сколько полей указателей содержит каждый узел двусвязного линейного списка?
   Каждый узел двунаправленного (двусвязного) линейного списка (ДЛС) содержит два поля указателей — на следующий и на предыдущий узлы.

8. Сколько полей указателей содержит каждый узел односвязного линейного списка?
   Каждый узел однонаправленного (односвязного) линейного списка (ОЛС) содержит одно поле указателя на следующий узел.

9. Основные действия, производимые над элементами двусвязного линейного списка.

   - Инициализация списка
   - Добавление узла в список
   - Удаление узла из списка
   - Удаление корня списка
   - Вывод элементов списка
   - Вывод элементов списка в обратном порядке
   - Взаимообмен двух узлов списка

Вариант 11

```
class Node {
  constructor(data, previous, next) {
    this.data = data;
    this.previous = previous;
    this.next = next;
  }
}
```

Рисунок 1 - Класс узла

```
class LinkedList {
  constructor(data) {
    this.header = new Node(data, null, null);
  }
```

Рисунок 2 - Класс двусвязного списка

```
find(data) {
  let currentNode = this.header; // корень
  while (currentNode.data ≠ data) {
    currentNode = currentNode.next; // пока не совпадёт с data
  }
  return currentNode;
}

getLast() {
  let currentNode = this.header;
  while (currentNode.next ≠= null) {
    // пока не будет указывать на пустой
    currentNode = currentNode.next;
  }
  return currentNode;
}

add(data) {
  let lastNode = this.getLast();
  let newNode = new Node(data, lastNode, null);
  lastNode.next = newNode;
}
```

```
remove(data) {
  let currentNode = this.find(data);
  if (currentNode.next ≠ null && currentNode.previous ≠ null) {
    currentNode.previous.next = currentNode.next;
    currentNode.next.previous = currentNode.previous;
    currentNode = null;
  } else if (currentNode.next == null) {
    currentNode.previous.next = null;
    currentNode = null;
  } else if (currentNode.prev == null) {
    // удалить корневой
    currentNode.next.previous = null;
    this.header = currentNode.next;
  }
}

removeSame() {
  // проверяем каждый элемент со всеми для поиска одинаковых элементов
  let currentNode = this.header;
  let nextNode = this.header;
  let x = 0;
  while (currentNode ≠ null) {
    nextNode = this.header;
    while (nextNode ≠ null) {
      if (nextNode ≠ currentNode && nextNode.data == currentNode.data) {
        this.remove(nextNode.data);
      }
      nextNode = nextNode.next;
    }
    currentNode = currentNode.next;
  }
}

display() {
  let currentNode = this.header;
  while (currentNode ≠ null) {
    console.log(currentNode.data);
    currentNode = currentNode.next;
  }
}
```

Рисунок 3 - Базовые методы

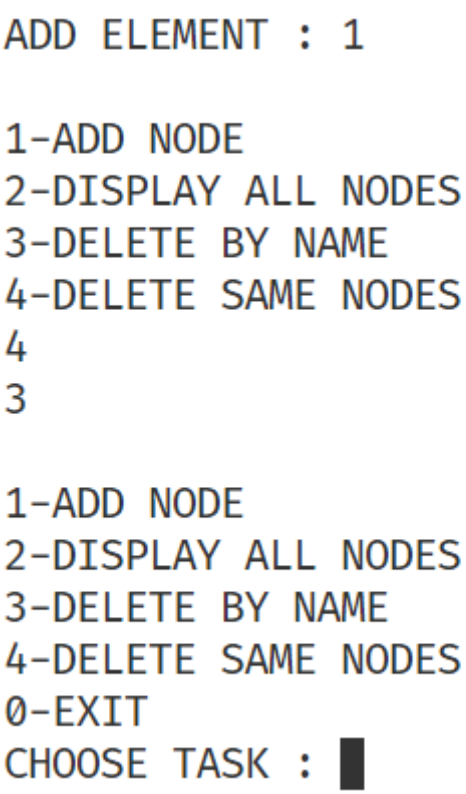

Рисунок 4 - Результат работы программы