МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
"МИРЭА - Российский технологический университет"
РТУ МИРЭА

Отчет по выполнению практического задания №7

Тема: Двунаправленный динамический список

Дисциплина: «Структуры и алгоритмы обработки данных»

Выполнил студент Антонов А.Д.

Группа ИКБО-01-20

Москва 2021

Содержание

1. Постановка задачи

Разработать многомодульную программу, которая демонстрирует выполнение всех операций, определенных вариантом, над линейным двунаправленным динамическим списком.

Требования к разработке.

1. Разработать структуру узла списка, структура информационной части узла определена вариантом. Для определения структуры узла списка, использовать тип struct. Сохранить определение структуры узла прототипы функций в заголовочном файле.

2. Разработать функции для выполнения операции над линейным двунаправленным динамическим списком:

- вывод списка в двух направлениях (слева направо и справа налево)
- поиск узла с заданным значением (операция должна возвращать указатель на узел с заданным значением)

3. Дополнительные операции над списком, указанные вариантом оформить в виде функций и включить в отдельный файл с расширением cpp. Подключите к этому файлу заголовочный файл с определением структуры узла.

4. Разработать программу, управляемую текстовым меню, и включить в меню демонстрацию выполнения всех операций задания и варианта.

5. Провести тестирование операций.

6. Оценить сложность алгоритма первой дополнительной операции для реализации линейного списка:

- на линейном динамическом списке
- на одномерном массиве.

7.Оформить отчет по разработке программы в соответствии с требованиями задания по однонаправленному списку.

Вариант 1. Номер зачетной книжки, номер группы, оценка. Дополнительные операции:

1) Вставить новый узел перед первым узлом с таким же ключом, если такого узда еще нет, то вставить перед первым узлом, у которого ключ больше.

2) Удалить узлы с указанным номером группы.

3) Сформировать новый список из исходного, включив в него узлы с оценкой неуд, исключив их при этом из исходного списка.

2. Операции над списком

2.1. Определение структуры узла двунаправленного списка

В отличие от однонаправленного динамического списка, двунаправленный содержит указатель на следующий узел и указатель на предыдущий узел.

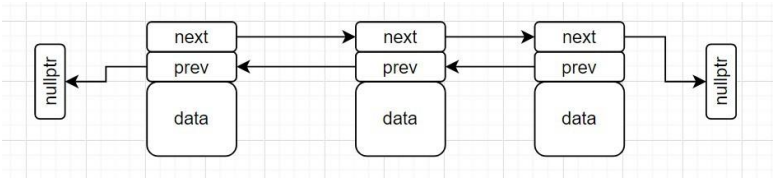

Рис. 1 Схема двунаправленного списка

Согласно варианту №1 в качестве информационной части узла списка используются поля: Номер зачетной книжки, номер группы, оценка.

Код реализации информационной части узла:

```
6   struct node {
7       int number_b{}, number_g{}, degree{};
8       node *prev = nullptr;
9       node *next = nullptr;
10  };
```

2.2. Вывод списка в прямом и обратном направлении

- Алгоритм:

1) Если количество узлов в списке не равно 0, то создаем указатель x на первый узел списка, и пока список не закончится, печатаем поля узла, на который указывает x и переходим к следующему узлу.

2) Если количество узлов в списке не равно 0, то создаем указатель x на последний узел списка, и пока список не закончится, печатаем поля узла, на который указывает x и переходим к предыдущему узлу.

- Схематические изображения:

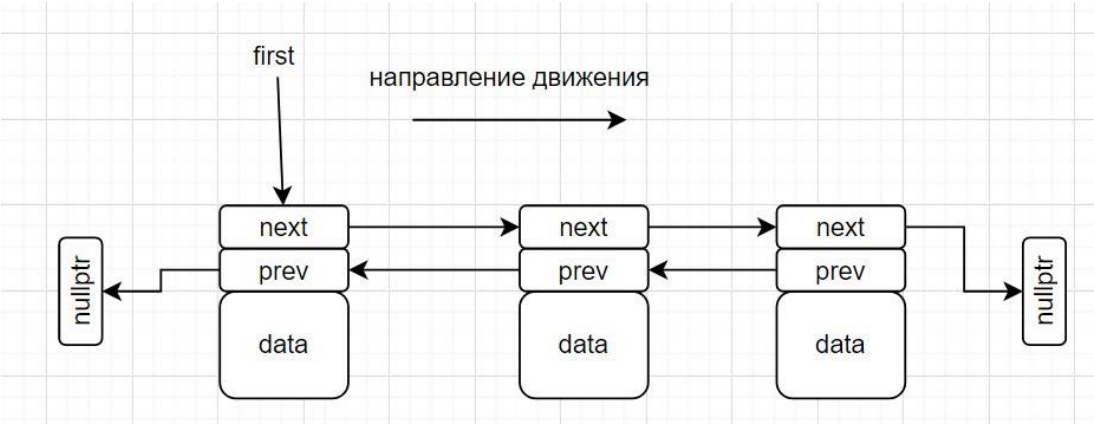

Рис. 2 Вывод списка в прямом направлении

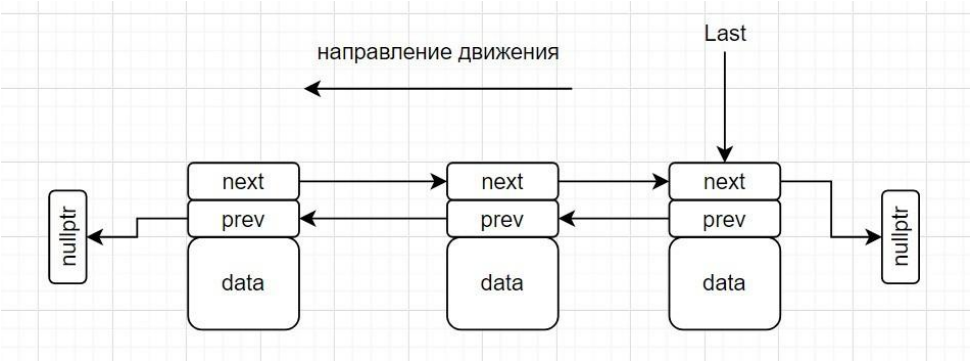

Рис. 3 Вывод списка в обратном направлении

- Код функций:

```
55  void list::print1() {
56      if (count != 0) {
57          node *x = first;
58          while (x != nullptr) {
59              cout << x->number_b << ' '
60                   << x->number_g << ' '
61                   << x->degree << '\n';
62              x = x->next;
63          }
64      }
65  }
```

```
67  void list::print2() {
68      if (count != 0) {
69          node *x = last;
70          while (x != nullptr) {
71              cout << x->number_b << ' '
72                   << x->number_g << ' '
73                   << x->degree << '\n';
74              x = x->prev;
75          }
76      }
77  }
```

• Тестирование:

```
Enter number of elements: 3
Enter values:
100-003 01 5
100-543 04 4
100-645 02 5
Functions:
1) Display ->
2) Display <-
3) Add node
4) Delete node
5) Create new list
6) Find node
Enter function number: 1
100-003 01 5
100-543 04 4
100-645 02 5
```

```
Enter number of elements: 3
Enter values:
100-003 01 5
100-543 04 4
100-645 02 5
Functions:
1) Display ->
2) Display <-
3) Add node
4) Delete node
5) Create new list
6) Find node
Enter function number: 2
100-645 02 5
100-543 04 4
100-003 01 5
```

Тестирование показало правильность работы функции.

2.3. Функция поиска узла с заданным значением

• Алгоритм:

Если номер зачетной книжки (узла) равен заданному значению (вводится с клавиатуры) возврат узла.

• Код функции:

```
node *list::findnode(int data) {
    node *node = first;
    while (node != nullptr) {
        if (node->number_b == data) {
            return node;
        }
        node = node->next;
    }
    cout << "Not found\n";
    return nullptr;
}
```

• Тестирование функции:

```
Enter number of elements: 3
Enter values:
100-003 01 5
100-543 04 4
100-645 02 5
Functions:
1) Display ->
2) Display <-
3) Add node
4) Delete node
5) Create new list
6) Find node
Enter function number: 6
Enter book number: 100-543
100-543 04 4
```

Тестирование показало правильность работы функции.

2.4. Функция добавления нового узла

- Алгоритм:

Вставить новый узел перед первым узлом с таким же ключом, если такого узла нет, то вставить перед первым узлом, у которого ключ больше.

- Схематическое изображение:

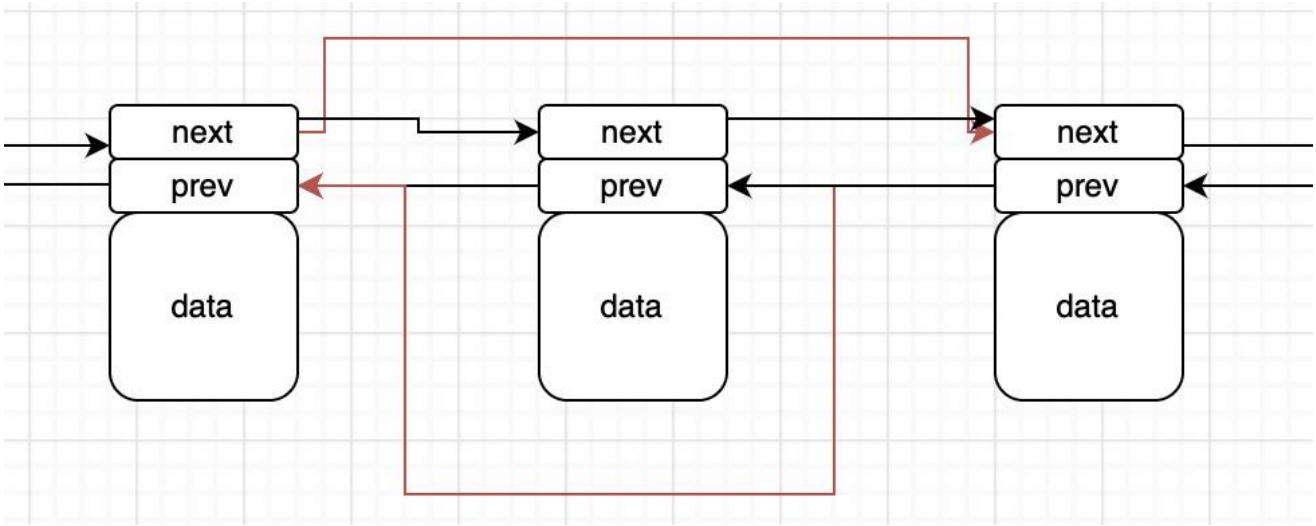

Рис. 7 Алгоритм перемены мест двух соседних узлов

- Код функции:

```
115  void list::push_back() {
116      cout << "Enter the number of book, number of group, degree" << '\n';
117      int number_b, number_g, degree;
118      cin >> number_b >> number_g >> degree;
119      node *x = new node;
120      x->number_b = number_b;
121      x->number_g = number_g;
122      x->degree = degree;
123      if (x->number_b == first->number_b) {
124          if (first->prev == nullptr) {}
125          else {
126              first->prev = x;
127              x->prev = nullptr;
128              x->next = first;
129              node *second = first->prev;
130              first->prev = x;
131              x->next = first;
132              x->prev = second;
133          }
134      }
135      count++;
136      print1();
137  }
```

- Тестирование:

```
Enter number of elements: 3
Enter values:
100-003 01 5
100-543 04 4
100-645 02 5
Functions:
1) Display ->
2) Display <-
3) Add node
4) Delete node
5) Create new list
6) Find node
Enter function number: 3
Enter new node: 100-243 06 4
100-003 01 5
100-543 04 4
100-243 06 4
100-645 02 5
```

Тестирование показало правильность работы функции.

2.5. Функция удаления узла с указанным значением

- Алгоритм:

Находим узел, удовлетворяющий условию, затем удаляем его.

- Схематическое изображение:

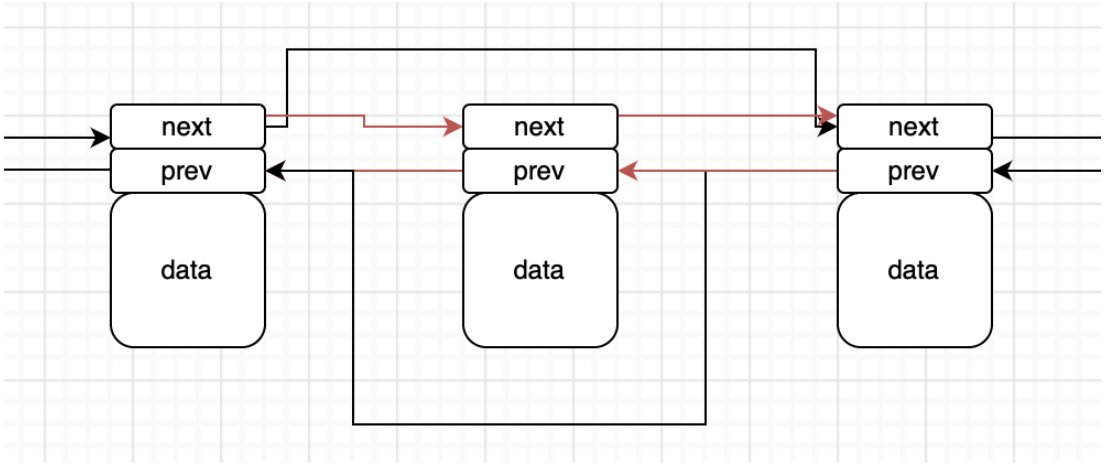

Рис. 9 Алгоритм перемены мест двух соседних узлов

- Код функции:

```
79  void list::del() {
80      int data;
81      cout << "Enter delete number of group";
82      cin >> data;
83      node *x = first;
84      while (x != nullptr) {
85          if (x->number_g == data) {
86              node *y = x->prev;
87              node *z = x->next;
88              if (y != nullptr && count != 1) y->next = z;
89              if (z != nullptr && count != 1) z->prev = y;
90              if (y == nullptr) first = z;
91              if (z == nullptr) last = y;
92              delete x;
93              count--;
94              if (z != nullptr)
95                  x = z;
96              else break;
97          }
98          x = x->next;
99      }
100     print1();
101 }
```

- Тестирование функции:

```
Enter number of elements: 3
Enter values:
100-003 01 5
100-543 04 4
100-645 02 5
Functions:
1) Display ->
2) Display <-
3) Add node
4) Delete node
5) Create new list
6) Find node
Enter function number: 4
Enter node: 100-543
100-003 01 5
100-645 02 5
```

Тестирование показало правильность работы функции.

2.6. Функция формирования списка

- Алгоритм:

Находим узел, удовлетворяющий условию, удаляем его из старого списка и вставляем в новый.

- Схематические изодражения:

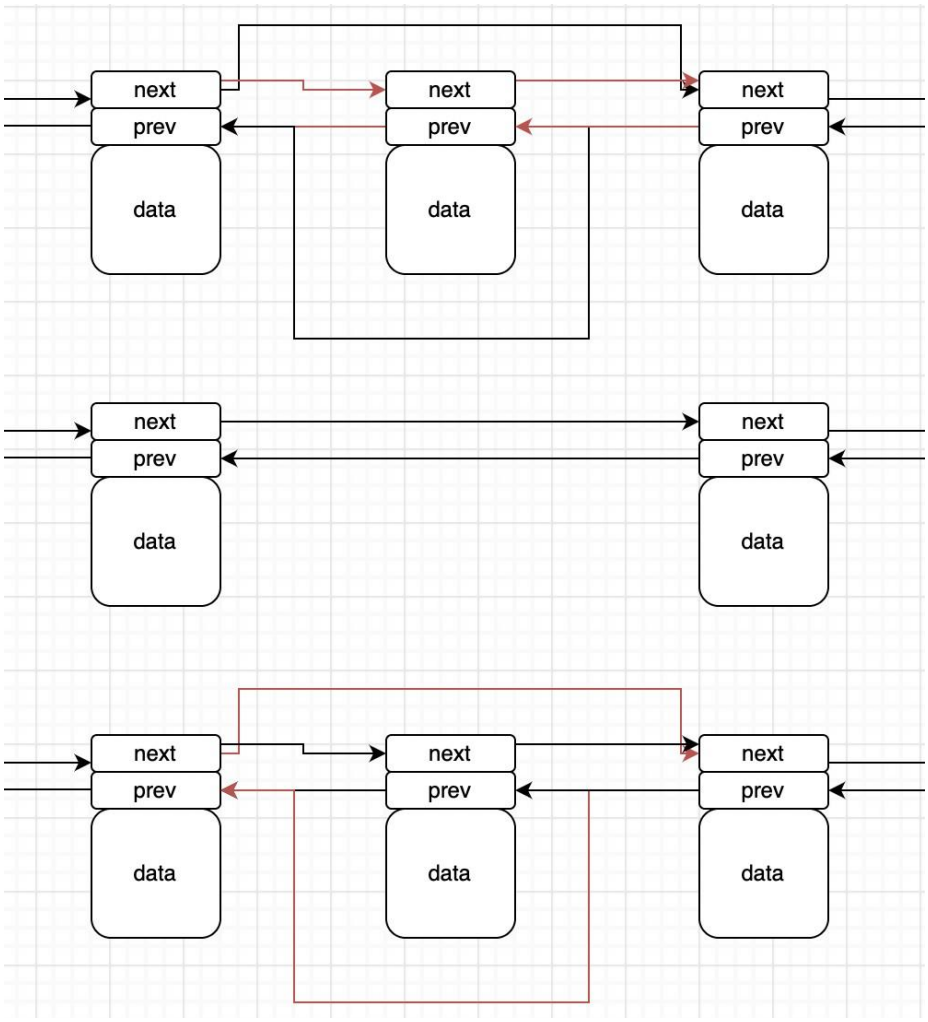

Рис. 11 Схема алгоритма

- Код функции:

```
void new_l(list lst1, list lst2) {
    node *x = lst1.first;
    while (x->next != nullptr) {
        if (x->degree == 2) {
            lst2.add_node(x->number_b, x->number_g, x->degree);
        }
        x = x->next;
    }
}
```

- Тестирование функции:

```
Enter number of elements: 3
Enter values:
100-003 01 5
100-543 04 4
100-645 02 5
Functions:
1) Display ->
2) Display <-
3) Add node
4) Delete node
5) Create new list
6) Find node
Enter function number: 5
Old list:
100-543 04 4
New list:
100-003 01 5
100-645 02 5
```

Тестирование показало правильность работы функции.

3. Код программы

```
12  class list {
13      node *first;
14      node *last;
15      int count;
16
17  public:
18      list() : first(nullptr), last(nullptr), count(0) {};
19      void print1();
20      void print2();
21      void create_list(int n);
22      void add_node(int number_b, int number_g, int degree);
23      void del();
24      void push_back();
25      void friend new_l(list lst1, list lst2);
26      node *findnode(int data);
27  };
```

```
29  void list::add_node(int number_b, int number_g, int degree) {
30      node *x = new node;
31      x->next = nullptr;
32      x->prev = nullptr;
33      x->number_b = number_b;
34      x->number_g = number_g;
35      x->degree = degree;
36      if (last != nullptr) {
37          last->next = x;
38          x->prev = last;
39      }
40      if (count == 0) first = last = x;
41      else
42          last = x;
43      count++;
44  }
```

```
46  void list::create_list(int n) {
47      cout << "Value:" << '\n';
48      int number_b, number_g, degree;
49      for (int i = 0; i < n; i++) {
50          cin >> number_b >> number_g >> degree;
51          add_node(number_b, number_g, degree);
52      }
53  }
```

```
186 int main() {
187     menu();
188 }
```

```
150  void menu() {
151      list l, l2;
152      int n;
153      cout << "Enter number of elements: ";
154      cin >> n;
155      l.create_list(n);
156      cout << "Functions:" << endl
157           << "1) Display ->" << endl
158           << "2) Display <-" << endl
159           << "3) Add node" << endl
160           << "4) Delete node" << endl
161           << "5) Create new list" << endl
162           << "6) Find node" << endl;
163      cout << "Enter function number: ";
164      int a;
165      cin >> a;
166      switch (a) {
167          case 1: { l.print1(); break; }
168          case 2: { l.print2(); break; }
169          case 3: { l.push_back(); break; }
170          case 4: { l.del(); break; }
171          case 5: { new_l(l, l2); break; }
172          case 6: {
173              cout << "Enter number of book: ";
174              int data;
175              cin >> data;
176              l.findnode(data);
177              break;
178          }
179          default: {
180              cout << "Choose the true number" << '\n';
181              break;
182          }
183      }
184  }
```

Выводы

В ходе практической работы был разработан двунаправленный динамический список, получены знания и практические навыки управления двунаправленным динамическим списком; реализованы необходимые функции взаимодействия со списком, включая задания индивидуального варианта, под управлением текстового меню. Каждая выполняющаяся над списком операция прошла тестирование.

Список информационных источников

1. Кораблин Ю.П., Сыромятников В.П., Скворцова Л.А. Учебно-методическое пособие Структуры и алгоритмы обработки данных, М.:МИРЭА, 2020
2. Никлаус Вирт Алгоритмы и структуры данных. Классика программирования – М.:ДМК Пресс, 2016. — 272 с.
3. Круз Р. Л. Структуры данных и проектирование программ / пер. с англ. — 3-е издание / Р.Л. Круз. – М.:Лаборатория знаний, 2017. — 768 с.
4. Альфред В. Ахо, Джон Э. Хопкрофт, Джеффри Д. Ульман. Структуры данных и алгоритмы М.:Вильямс, 2016 — 400 с.