

1. Цель работы

Целью работы является изучение структур данных «линейный список» и «циклический список», а также получение практических навыков их реализации.

2. Задание

Реализовать структуры данных «линейный список» и «циклический список» в соответствии с заданным вариантом. Дополнительно программа должна осуществлять следующие операции:

- 1) Добавление/удаление элемента в список (с клавиатуры);
- 2) Вывод исходного и результирующего списков на экран;
- 3) Если списки являются многочленами, в выводе должна быть отражена степень каждого элемента.

№ ва р.	Задача	Вид списка
1	Дана последовательность повторяющихся целых чисел a_1, a_2, \dots, a_n . Получить последовательность k_1, k_2, \dots, k_m , содержащую повторяющиеся в исходной последовательности элементы (c) в порядке убывания частоты их повторения в исходной последовательности (count). Неповторяющиеся элементы не включать. $k_1 = \max(\text{count}(c)), k_2 = \min(\max(\text{count}(c), k_1)) \dots$	Линейный односвязный

3. Описание созданных функций

Для реализации задания были использованы следующие функции:

Имя: Addelem.

Назначение: добавление элемента в список.

Входные данные:

- *list – указатель на позицию в списке;
- value – переменная, хранящая в себе значение, которое нужно поместить в список;

Выходные данные:

Побочный эффект: отсутствует.

Имя: print.

Назначение: печать содержимого списка в консоль.

Входные данные:

- *list – указатель на позицию в списке;

Выходные данные:

Побочный эффект: отсутствует.

Имя: removefirst.

Назначение: удаление 1 элемента списка.

Входные данные:

- *list – указатель на позицию в списке;

Выходные данные:

Побочный эффект: каждый следующий элемент в списке за удалённым будет считаться первым.

Имя: input.

Назначение: заполнение списка.

Входные данные:

- *list – указатель на позицию в списке;

Выходные данные: возвращает число элементов в списке

Побочный эффект: отсутствует.

Имя: sort.

Назначение: перегруженная функция для сортировки исходного массива.

Входные данные:

- *a– массив int размером n;
- n – число элементов в списке

Выходные данные:

Побочный эффект: отсутствует

Имя: sort.

Назначение: перегруженная функция для сортировки повторяющихся значение по убыванию.

Входные данные:

- vc– переменная типа структуры VC , хранящая в себе значения повторяющихся элементов;
- n – число элементов в списке

Выходные данные:

Побочный эффект: отсутствует

Имя: printmenu.

Назначение: печать пунктов меню.

Входные данные:

Выходные данные:

Побочный эффект: отсутствует

4. Листинг программы

```
1. #include <iostream>
2. #include <cstdlib>
3. #include <ctime>

4. using namespace std;

5. struct Elem
6. {
7.     int value; // переменная хранящая значение элемента
8.     Elem *next; // указатель на следующий элемент списка
9. };

10. struct List
11. {
12.     Elem *begin; // указатель на начало списка
13.     Elem *end; // указатель на конец списка
14. List()
15. {
16.     // присваиваем указателям значение
17.     begin = NULL;
18.     end = NULL;
19. }
20. };

21. void Addelem(List *list, int value) // добавление элемента в список
22. {
23.     Elem * newelem = new Elem; // здесь new это создание переменной типа Elem

24.     newelem->value = value; // присваиваем переменной значение value
25.     newelem->next = NULL; // указатель на следующий элемент теперь смотрит на
        NULL
26.     if (list->end != NULL) // проверка если указатель на конец списка не равен NULL,
        то
27.     {
28.         list->end->next = newelem; // передвигаем указатель на newelem
29.     }

30.     list->end = newelem;

31.     if (list->begin == NULL) //проверка если указатель на начало списка не равен NULL,
        то
32.     {
33.         list->begin = newelem; // указатель на начала списка будет сдвинут на newelem
34.     }
35. }

36. void print(List *list) // функция для печати списка
37. {
38.     for (Elem *p = list->begin; p != NULL; p = p->next) // выводим элементы начиная от
        указателя на начало списка до того момента пока p=NULL
39.     {
40.         cout << p->value << "->";

41.     }
42.     cout << "Null\n";
43. }

44. void removefirst(List * list) // функция для удаления первого элемента списка
45. {
46.     Elem *tmp; // создание буферной переменной типа Elem
```

```

47. if (list->end == NULL) //проверка если указатель на конец списка ссылается на
    NULL, то
48. {
49. return; // выходим
50. }

51. tmp = list->begin->next; //переменной-буферу присваиваем значение следующего
    элемента
52. delete list->begin; // удаляем указатель на начало списка
53. list->begin = tmp; // присваиваем указатель начала списка переменной записанной до
    этого в tmp
54. }

55. int input(List *list)
56. {
57. int n, x;
58. cout << "Введите количество элементов списка: ";
59. cin >> n;

60. srand(time(NULL));
61. for (int i = 0; i < n; i++)
62. {
63. //cin >> x; // для ручного ввода
64. x = rand() % 10; // рандомная генерация элементов списка
65. Addelem(list, x);
66. }
67. return n;
68. }

69. struct VC
70. {
71. int value;
72. int count;
73. };

74. void sort(int *a, int n);

75. void sort(VC *vc, int n);

76. void printmenu();

77. int main()
78. {
79. setlocale(LC_ALL, "Russian");

80. int choise = 0;
81. int index = 0;

82. List A;
83. List B;
84. int n = input(&A);
85. print(&A);

86. int *a = new int[n];

87. // Create array from list A
88. int i = 0;
89. for (Elem *p = A.begin; p != NULL; p = p->next)
90. {
91. a[i++] = p->value;
92. }

93. sort(a, n);

94. VC *vc = new VC[n];

```

```

95. do
96. {
97. cout << "Введите номер пункта меню: " << endl;
98. printmenu();
99. cin >> choise;

100.     switch (choise)
101.     {
102.     case 1:
103.     {

104.         for (int i = 0; i < n; i++)
105.         {
106.             int v = a[i];
107.             int c = 1;
108.             while (i < n - 1 && a[i + 1] == v)
109.             {
110.                 i++;
111.                 c++;
112.             }
113.             if (c > 1)
114.             {
115.                 vc[index].value = v;
116.                 vc[index].count = c;
117.                 index++;
118.             }
119.         }

120.         // Create result list2
121.         for (int i = 0; i < index; i++)
122.         {
123.             Addelem(&B, vc[i].value);
124.         }
125.         print(&B);

126.         // Sort vc array by count decrease
127.         sort(vc, index);
128.     }

129.     case 2:
130.     {
131.         //Удаление элементов списка
132.         while (A.begin != NULL)
133.             removefirst(&A);

134.         while (B.begin != NULL)
135.             removefirst(&B);
136.     }

137.     case 9:break;
138.     }

139.     }
140.     while (choise !=9);

141.     // Free memory
142.     delete[]a;
143.     delete[]vc;

144.     _CrtSetReportMode(_CRT_WARN, _CRTDBG_MODE_FILE);
145.     _CrtSetReportFile(_CRT_WARN, _CRTDBG_FILE_STDOUT);
146.     _CrtSetReportMode(_CRT_ERROR, _CRTDBG_MODE_FILE);
147.     _CrtSetReportFile(_CRT_ERROR, _CRTDBG_FILE_STDOUT);

```

```
148.     _CrtSetReportMode(_CRT_ASSERT, _CRTDBG_MODE_FILE);
149.     _CrtSetReportFile(_CRT_ASSERT, _CRTDBG_FILE_STDOUT);
150.     _CrtDumpMemoryLeaks();

151.     system("pause");
152.     return 0;
153. }

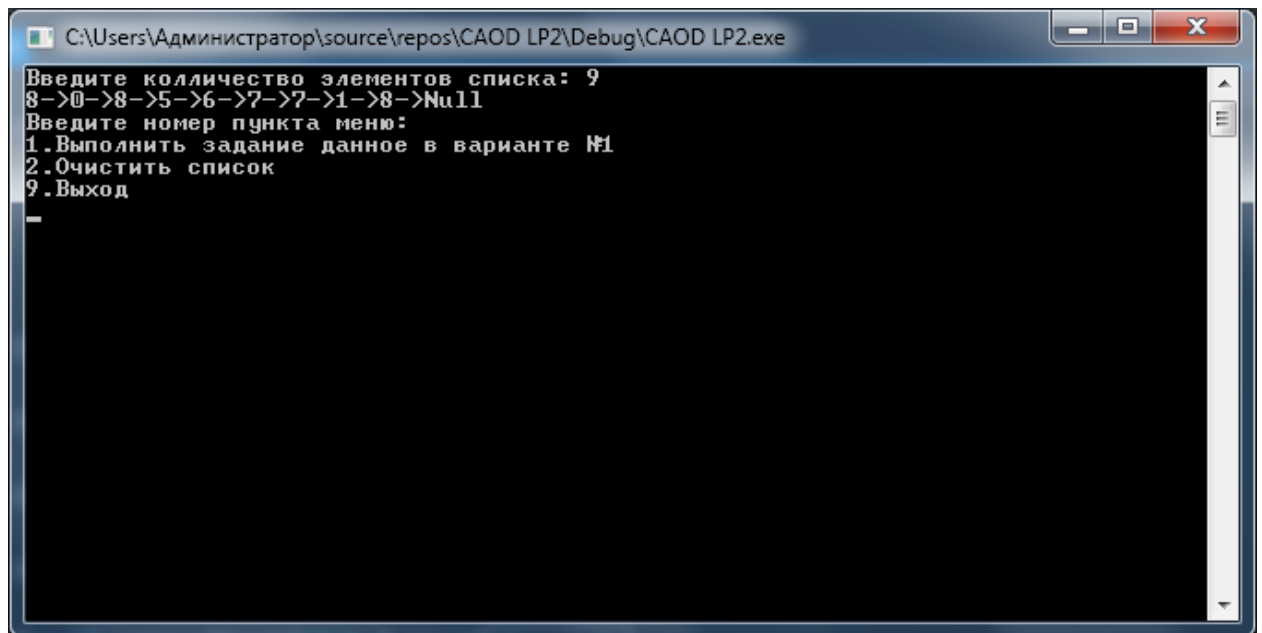
154.     void sort(int *a, int n)
155.     {
156.         for (int i = 0; i < n - 1; i++)
157.             for (int j = i + 1; j < n; j++)
158.                 if (a[i] > a[j])
159.                 {
160.                     int tmp = a[i];
161.                     a[i] = a[j];
162.                     a[j] = tmp;
163.                 }
164.     }

165.     void sort(VC *vc, int n)
166.     {
167.         for (int i = 0; i < n - 1; i++)
168.             for (int j = i + 1; j < n; j++)
169.                 if (vc[i].count < vc[j].count)
170.                 {
171.                     VC tmp = vc[i];
172.                     vc[i] = vc[j];
173.                     vc[j] = tmp;
174.                 }
175.     }

176.     void printmenu()
177.     {
178.         cout << "1.Выполнить задание данное в варианте №1" << endl;
179.         cout << "2.Очистить список" << endl;
180.         cout << "9.Выход" << endl;
181.
182.     }
```

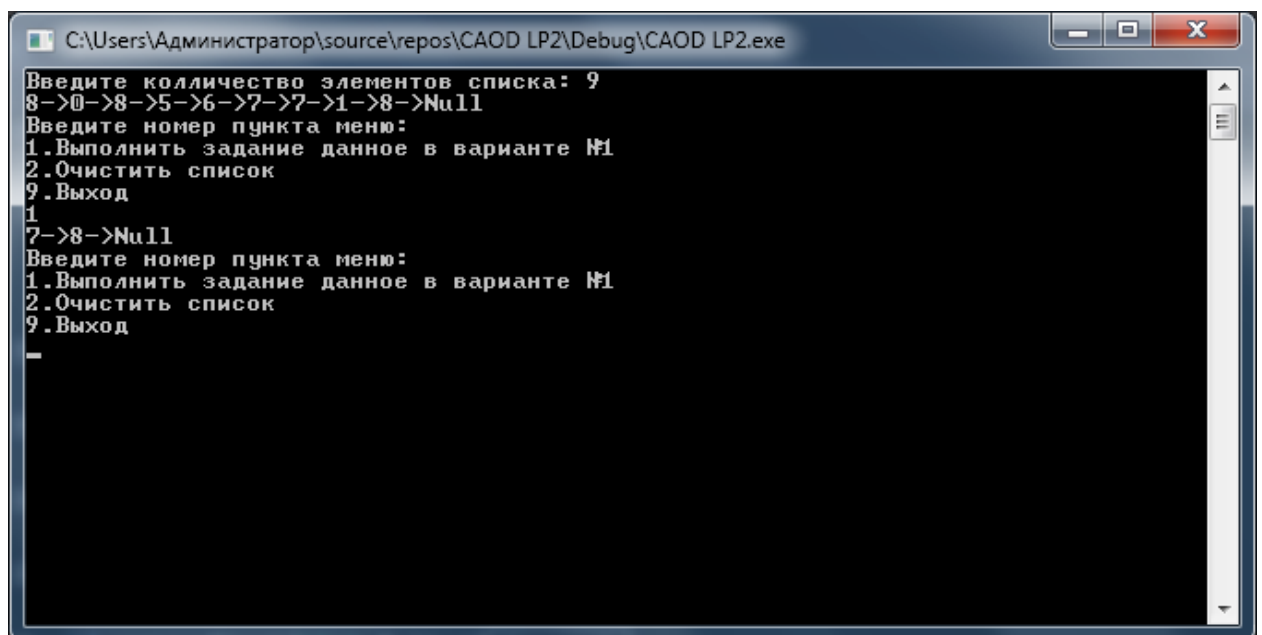
5. Пример выполнения программы

Ниже приведён пример выполнения программы



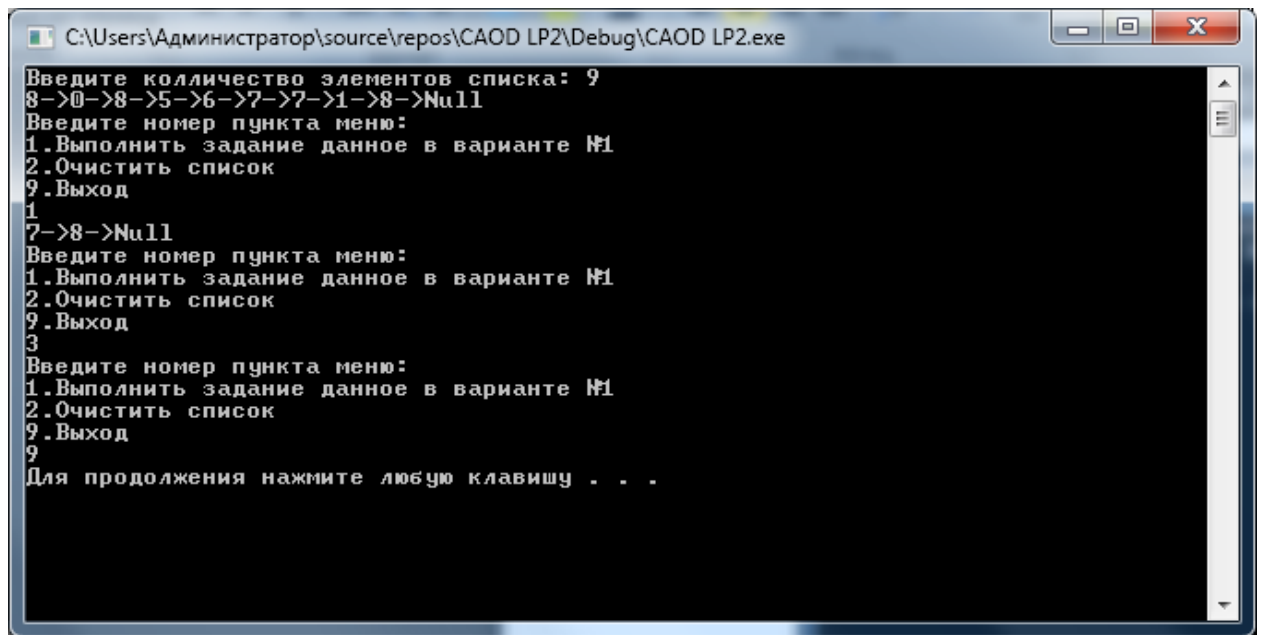
```
C:\Users\Администратор\source\repos\CAOD LP2\Debug\CAOD LP2.exe
Введите количество элементов списка: 9
8->0->8->5->6->7->7->1->8->Null
Введите номер пункта меню:
1.Выполнить задание данное в варианте №1
2.Очистить список
9.Выход
-
```

Рис 1. Пример выполнения программы



```
C:\Users\Администратор\source\repos\CAOD LP2\Debug\CAOD LP2.exe
Введите количество элементов списка: 9
8->0->8->5->6->7->7->1->8->Null
Введите номер пункта меню:
1.Выполнить задание данное в варианте №1
2.Очистить список
9.Выход
1
7->8->Null
Введите номер пункта меню:
1.Выполнить задание данное в варианте №1
2.Очистить список
9.Выход
-
```

Рис 2. Пример выполнения программы



```
C:\Users\Администратор\source\repos\CAOD LP2\Debug\CAOD LP2.exe
Введите количество элементов списка: 9
8->0->8->5->6->7->7->1->8->Null
Введите номер пункта меню:
1.Выполнить задание данное в варианте №1
2.Очистить список
9.Выход
1
7->8->Null
Введите номер пункта меню:
1.Выполнить задание данное в варианте №1
2.Очистить список
9.Выход
3
Введите номер пункта меню:
1.Выполнить задание данное в варианте №1
2.Очистить список
9.Выход
9
Для продолжения нажмите любую клавишу . . .
```

Рис 3. Пример выполнения программы