Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение

высшего образования

«**Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные

системы»

**ОТЧЁТ**

**по лабораторной работе №11.2**

Дисциплина: «основы алгоритмизации и программирования»

Тема: Двунаправленный список

Вариант 17

Выполнил:

Студент группы ИВТ-20-2б

Сафронов Владислав Владиславович

Проверила:

Доцент кафедры ИТАС

Полякова О.А.

**Пермь, 2021**

**Цель задачи**

Получить практические навыки работы с двунаправленными списками.

**Постановка задачи**

Реализовать двунаправленный список. Тип информационного поля char\*. Добавить в список элемент с заданным номером.

**Анализ задачи**

**Какие типы данных будут использованы:**

* char
* char\*
* int

**Какие с этими данными выполняются действия:**

* int – сравнение, математические операции;
* char\* - создание массива char, добавление элементов

**В каком виде эти данные будут представлены:**

**Реализация через структуры**

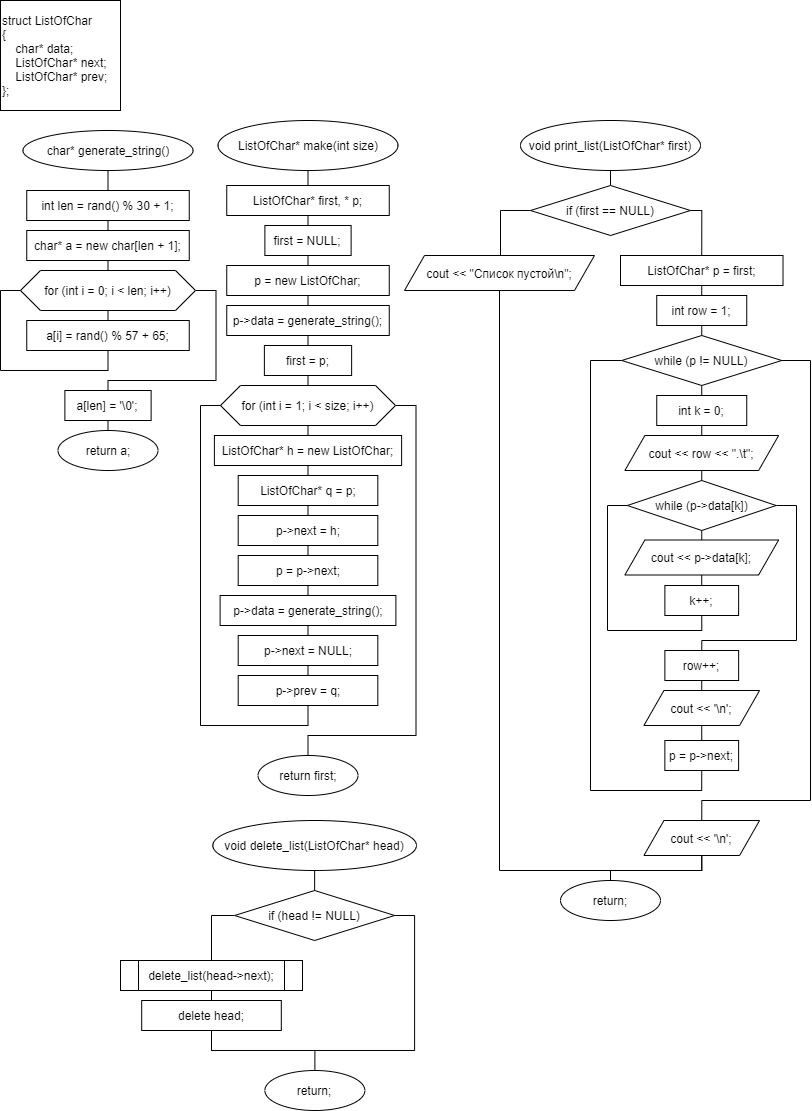
1. Структура ListOfChar реализует двунаправленный список. Структура имеет 3 поля: data, в котором хранится массив символов, а также указатели на предыдущий и следующий элемент списка.
2. struct ListOfChar
3. {
4. char\* data;
5. ListOfChar\* next;
6. ListOfChar\* prev;
7. };
8. Функция generate\_string генерирует случайную строку с помощью ДСЧ. Возвращает указатель на массив символов char.
9. char\* generate\_string()
10. {
11. int len = rand() % 30 + 1;
12. char\* a = new char[len + 1];
13. for (int i = 0; i < len; i++)
14. {
15. a[i] = rand() % 57 + 65;
16. }
17. a[len] = '\0';
19. return a;
20. }
21. Функция make для создания двунаправленного списка. В качестве входного параметра функция принимает размер двунаправленного списка. Значения элементов списка генерируются с помощью ДСЧ. Функция возвращает указатель на первый элемент сгенерированного списка.
22. ListOfChar\* make(int size)
23. {
24. ListOfChar\* first, \* p;
25. first = NULL;
26. p = new ListOfChar;
27. p->data = generate\_string();
28. first = p;
29. for (int i = 1; i < size; i++)
30. {
31. ListOfChar\* h = new ListOfChar;
32. ListOfChar\* q = p;
33. p->next = h;
34. p = p->next;
35. p->data = generate\_string();
36. p->next = NULL;
37. p->prev = q;
38. }
39. return first;
40. }
41. Функция print\_list для вывода двунаправленного списка в консоль.
42. void print\_list(ListOfChar\* first)
43. {
44. if (first == NULL) cout << "Список пустой\n";
45. else
46. {
47. ListOfChar\* p = first;
48. int row = 1;
49. while (p != NULL)
50. {
51. int k = 0;
52. cout << row << ".\t";
53. while (p->data[k])
54. {
55. cout << p->data[k];
56. k++;
57. }
58. row++;
59. cout << '\n';
60. p = p->next;
61. }
62. cout << '\n';
63. }
64. }
65. Функция delete\_list для удаления всех элементов двунаправленного списка. Функция вызывается рекурсивно.
66. void delete\_list(ListOfChar\* head)
67. {
68. if (head != NULL)
69. {
70. delete\_list(head->next);
71. delete head;
72. }
73. }
74. Функция append\_to\_list для добавления элемента в список на заданную позицию. В качестве входных параметров функция принимает указатель на элемент, который нужно добавить в список и размер списка. Пользователь вводит номер, на который нужно добавить элемент. Если ввод корректен, то генерируется новый элемент списка и добавляется на заданную позицию.
75. void append\_to\_list(ListOfChar\*& first, int size)
76. {
77. int pos, k = 1;
78. ListOfChar\* p = first;
79. cout << "Введите номер строки для добавления (1-" << size + 1 << "): ";
80. cin >> pos;
81. while (pos < 1 || pos > (size + 1))
82. {
83. cout << "Номер введен некорректно.\n";
84. cout << "\nВведите номер строки для добавления (1-" << size + 1 << "): ";
85. cin >> pos;
86. }
87. if (pos == 1)
88. {
89. ListOfChar\* h = new ListOfChar;
90. h->data = generate\_string();
91. h->prev = NULL;
92. h->next = p;
93. first = h;
94. }
95. else
96. {
97. while (k < (pos - 1))
98. {
99. p = p->next;
100. k++;
101. }
102. ListOfChar\* h = new ListOfChar;
103. h->data = generate\_string();
104. h->prev = p;
105. if (p->next != NULL)
106. {
107. h->next = p->next;
108. p->next = h;
109. }
110. else
111. {
112. p->next = h;
113. h->next = NULL;
114. }
115. }
116. }

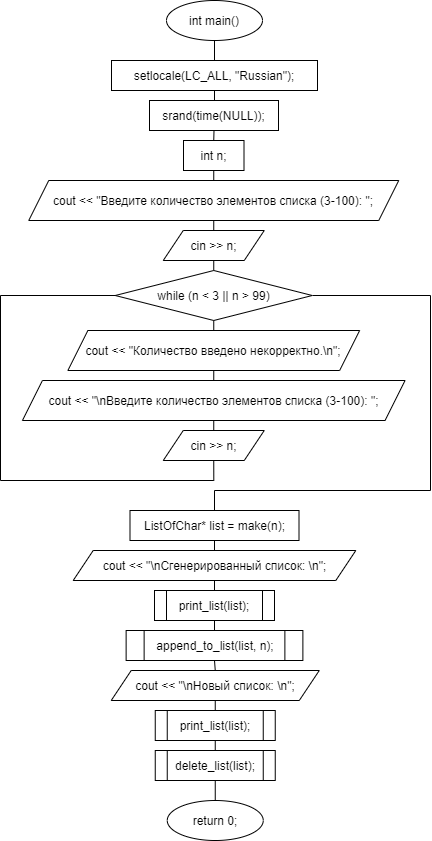
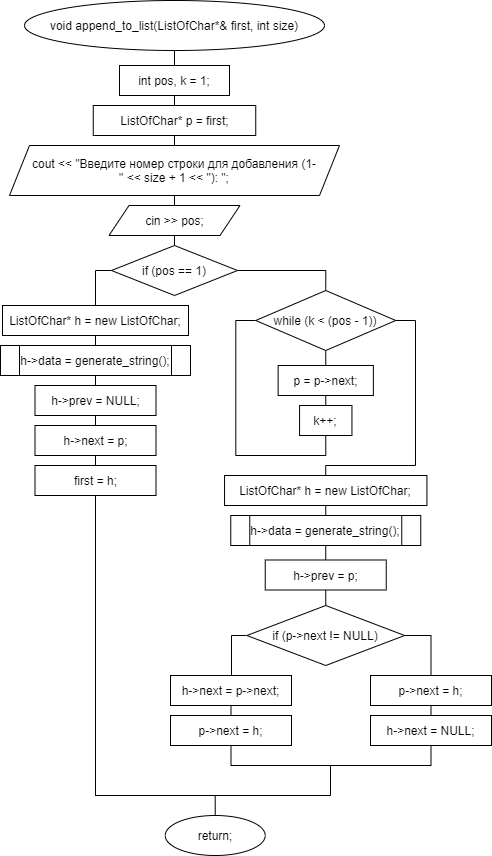
**Реализация через STL**

1. Глобальный список элементов char\*.
2. list<char\*> charlist;
3. Функция generate\_string для генерации строки. Аналогична с функицей генерации строки в реализации через структуры.
4. char\* generate\_string()
5. {
6. int len = rand() % 30 + 1;
7. char\* a = new char[len + 1];
8. for (int i = 0; i < len; i++)
9. {
10. a[i] = rand() % 57 + 65;
11. }
12. a[len] = '\0';
14. return a;
15. }
16. Функция make генерирует и добаляет элементы в список. В качестве входного параметра принимает размер списка.
17. void make(int size)
18. {
19. for (int i = 0; i < size; i++)
20. {
21. charlist.push\_back(generate\_string());
22. }
23. }
24. Функция add для добавления элемента на заданную позицию
25. void add(int size)
26. {
27. int pos;
28. cout << "Введите номер строки для добавления (1-" << size + 1 << "): ";
29. cin >> pos;
30. while (pos < 1 || pos >(size + 1))
31. {
32. cout << "Номер введен некорректно.\n";
33. cout << "\nВведите номер строки для добавления (1-" << size + 1 << "): ";
34. cin >> pos;
35. }
36. list<char\*> ::iterator it;
37. it = charlist.begin();
38. advance(it, pos - 1);
39. list<char\*> additlist;
40. additlist.push\_back(generate\_string());
41. charlist.insert(it, additlist.begin(), additlist.end());
42. }

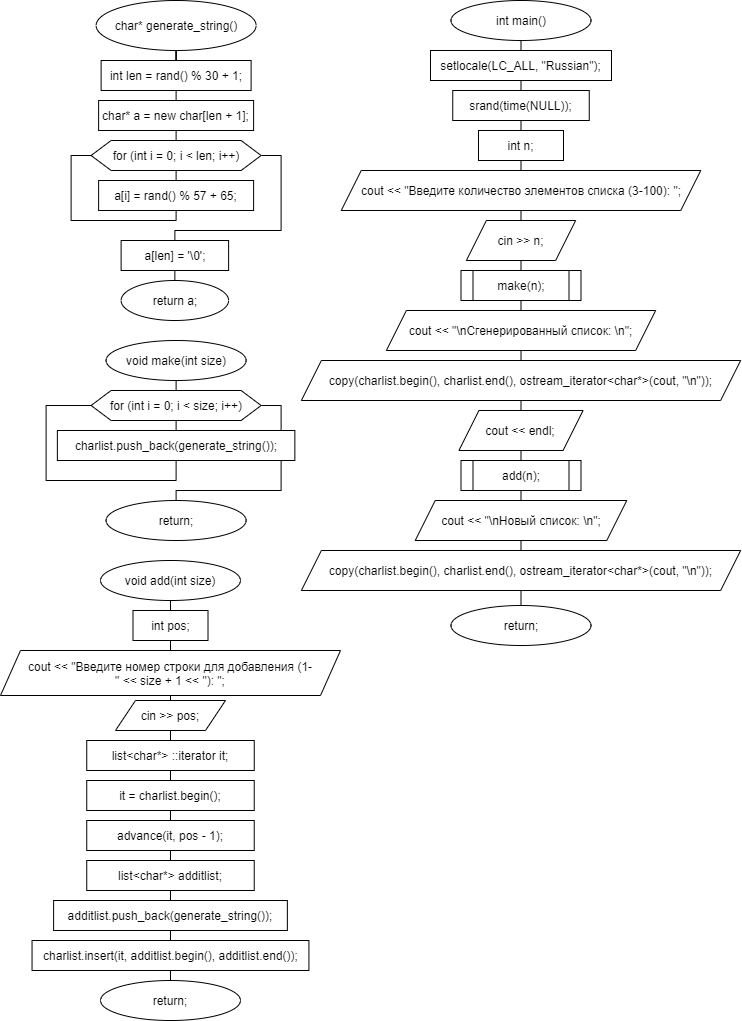
**Блок-схема программы**

**Реализация через структуры**

****



**Реализация через STL**

****

**Код программы**

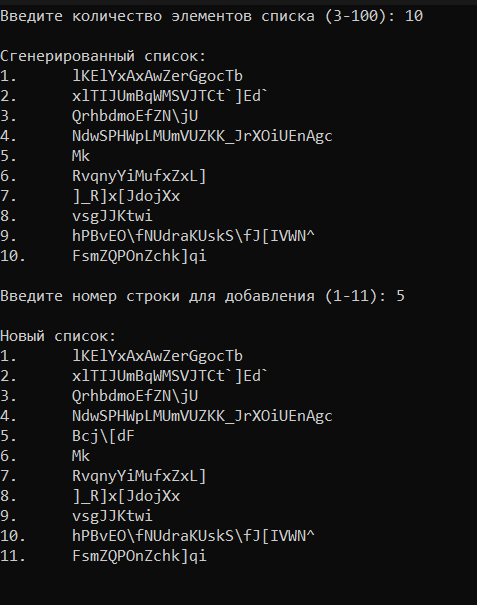
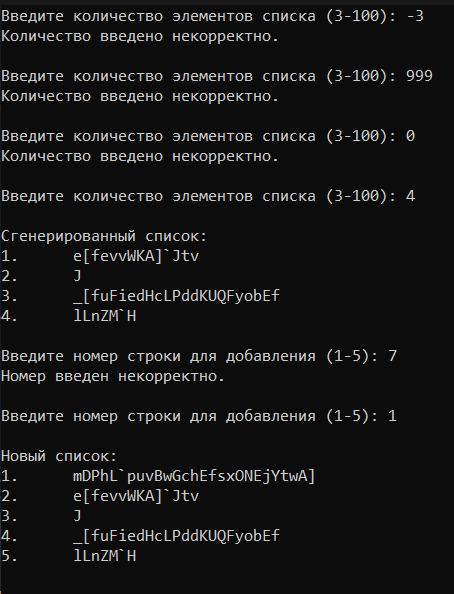
**Реализация через структуры**

1. #include <iostream>
2. #include <ctime>
3. using namespace std;
5. struct ListOfChar
6. {
7. char\* data;
8. ListOfChar\* next;
9. ListOfChar\* prev;
10. };
12. char\* generate\_string()
13. {
14. int len = rand() % 30 + 1;
15. char\* a = new char[len + 1];
16. for (int i = 0; i < len; i++)
17. {
18. a[i] = rand() % 57 + 65;
19. }
20. a[len] = '\0';
22. return a;
23. }
25. ListOfChar\* make(int size)
26. {
27. ListOfChar\* first, \* p;
28. first = NULL;
29. p = new ListOfChar;
30. p->data = generate\_string();
31. first = p;
32. for (int i = 1; i < size; i++)
33. {
34. ListOfChar\* h = new ListOfChar;
35. ListOfChar\* q = p;
36. p->next = h;
37. p = p->next;
38. p->data = generate\_string();
39. p->next = NULL;
40. p->prev = q;
41. }
42. return first;
43. }
45. void print\_list(ListOfChar\* first)
46. {
47. if (first == NULL) cout << "Список пустой\n";
48. else
49. {
50. ListOfChar\* p = first;
51. int row = 1;
52. while (p != NULL)
53. {
54. int k = 0;
55. cout << row << ".\t";
56. while (p->data[k])
57. {
58. cout << p->data[k];
59. k++;
60. }
61. row++;
62. cout << '\n';
63. p = p->next;
64. }
65. cout << '\n';
66. }
67. }
69. void delete\_list(ListOfChar\* head)
70. {
71. if (head != NULL)
72. {
73. delete\_list(head->next);
74. delete head;
75. }
76. }
78. void append\_to\_list(ListOfChar\*& first, int size)
79. {
80. int pos, k = 1;
81. ListOfChar\* p = first;
82. cout << "Введите номер строки для добавления (1-" << size + 1 << "): ";
83. cin >> pos;
84. while (pos < 1 || pos > (size + 1))
85. {
86. cout << "Номер введен некорректно.\n";
87. cout << "\nВведите номер строки для добавления (1-" << size + 1 << "): ";
88. cin >> pos;
89. }
90. if (pos == 1)
91. {
92. ListOfChar\* h = new ListOfChar;
93. h->data = generate\_string();
94. h->prev = NULL;
95. h->next = p;
96. first = h;
97. }
98. else
99. {
100. while (k < (pos - 1))
101. {
102. p = p->next;
103. k++;
104. }
105. ListOfChar\* h = new ListOfChar;
106. h->data = generate\_string();
107. h->prev = p;
108. if (p->next != NULL)
109. {
110. h->next = p->next;
111. p->next = h;
112. }
113. else
114. {
115. p->next = h;
116. h->next = NULL;
117. }
118. }
119. }
121. int main() {
122. setlocale(LC\_ALL, "Russian");
123. srand(time(NULL));
124. int n;
125. cout << "Введите количество элементов списка (3-100): ";
126. cin >> n;
127. while (n < 3 || n > 99) {
128. cout << "Количество введено некорректно.\n";
129. cout << "\nВведите количество элементов списка (3-100): ";
130. cin >> n;
131. }
132. ListOfChar\* list = make(n);
133. cout << "\nСгенерированный список: \n";
134. print\_list(list);
135. append\_to\_list(list, n);
136. cout << "\nНовый список: \n";
137. print\_list(list);
138. delete\_list(list);
139. return 0;
140. }

**Реализация через STL**

1. #include <iostream>
2. #include <ctime>
3. #include <iterator>
4. #include <list>
5. using namespace std;
7. list<char\*> charlist;
9. char\* generate\_string()
10. {
11. int len = rand() % 30 + 1;
12. char\* a = new char[len + 1];
13. for (int i = 0; i < len; i++)
14. {
15. a[i] = rand() % 57 + 65;
16. }
17. a[len] = '\0';
19. return a;
20. }
22. void make(int size)
23. {
24. for (int i = 0; i < size; i++)
25. {
26. charlist.push\_back(generate\_string());
27. }
28. }
30. void add(int size)
31. {
32. int pos;
33. cout << "Введите номер строки для добавления (1-" << size + 1 << "): ";
34. cin >> pos;
35. while (pos < 1 || pos >(size + 1))
36. {
37. cout << "Номер введен некорректно.\n";
38. cout << "\nВведите номер строки для добавления (1-" << size + 1 << "): ";
39. cin >> pos;
40. }
41. list<char\*> ::iterator it;
42. it = charlist.begin();
43. advance(it, pos - 1);
44. list<char\*> additlist;
45. additlist.push\_back(generate\_string());
46. charlist.insert(it, additlist.begin(), additlist.end());
47. }
49. int main()
50. {
51. setlocale(LC\_ALL, "Russian");
52. srand(time(NULL));
54. int n;
56. cout << "Введите количество элементов списка (3-100): ";
57. cin >> n;
58. while (n < 3 || n > 99) {
59. cout << "Количество введено некорректно.\n";
60. cout << "\nВведите количество элементов списка (3-100): ";
61. cin >> n;
62. }
63. make(n);
64. cout << "\nСгенерированный список: \n";
65. copy(charlist.begin(), charlist.end(), ostream\_iterator<char\*>(cout, "\n"));
66. cout << endl;
67. add(n);
68. cout << "\nНовый список: \n";
69. copy(charlist.begin(), charlist.end(), ostream\_iterator<char\*>(cout, "\n"));
70. return 0;
71. }

**Скриншоты результатов работы программы**

****