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 №11.4**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: Очередь

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 очередями.

**Постановка задачи**

Реализовать очередь. Тип информационного поля char\*. Добавить в очередь 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 Queue реализует очередь. Структура имеет 2 поля: data, в котором хранится массив символов, и указатель на следующий элемент очереди.
2. struct Queue
3. {
4. char\* data;
5. Queue\* next;
6. };
7. Функция generate\_string генерирует случайную строку с помощью ДСЧ. Возвращает указатель на массив символов char.
8. char\* generate\_string()
9. {
10. int len = rand() % 30 + 1;
11. char\* a = new char[len + 1];
12. for (int i = 0; i < len; i++)
13. {
14. a[i] = rand() % 57 + 65;
15. }
16. a[len] = '\0';
18. return a;
19. }
20. Функция make для создания очереди. В качестве входных параметров функция принимает размер очереди и указатели на первый и последний элемент очереди. Значения элементов списка генерируются с помощью ДСЧ. Функция возвращает указатель на первый элемент сгенерированной очереди.
21. Queue\* make(int n, Queue\*& first, Queue\*& last)
22. {
23. if (n == 0) return NULL;
24. Queue\* p = new Queue;
25. p->data = generate\_string();
26. p->next = NULL;
27. first = p;
28. last = p;
29. for (int i = 1; i < n; i++)
30. {
31. Queue\* h = new Queue;
32. h->data = generate\_string();
33. h->next = p;
34. last = h;
35. p = last;
36. }
37. return first;
38. }
39. Функция print\_queue для вывода элементов очереди в консколь.
40. void print\_queue(Queue\* last, int size)
41. {
42. Queue\* p = last;
43. int k = size;
44. while (p != NULL)
45. {
46. int i = 0;
47. cout << k << ".\t";
48. while (p->data[i])
49. {
50. cout << p->data[i];
51. i++;
52. }
53. cout << endl;
54. k--;
55. p = p->next;
56. }
57. cout << endl;
58. }
59. Функция pop, которая удаляет первый элемент очереди и возвращает его значение.
60. char\* pop(Queue\*& first, Queue\*& last)
61. {
62. Queue\* p = last;
63. int k = 0;
64. while (p != NULL)
65. {
66. p = p->next;
67. k++;
68. }
69. p = last;
70. if (k == 1)
71. {
72. char\* str = last->data;
73. delete p;
74. last = NULL;
75. first = NULL;
76. return str;
77. }
78. else
79. {
80. while (p->next->next != NULL)
81. {
82. p = p->next;
83. }
84. Queue\* t = p;
85. char\* str = p->next->data;
86. p = p->next;
87. t->next = NULL;
88. first = t;
89. delete p;
90. return str;
91. }
92. }
93. Функция push для добавления элементов в очередь.
94. Queue\* push(char\* n, Queue\*& last)
95. {
96. Queue\* p = new Queue;
97. p->data = n;
98. p->next = last;
99. last = p;
100. return last;
101. }

**Реализация через STL**

1. Функция generate\_string для генерации строки. Аналогична с функицей генерации строки в реализации через структуры
2. char\* generate\_string()
3. {
4. int len = rand() % 30 + 1;
5. char\* a = new char[len + 1];
6. for (int i = 0; i < len; i++)
7. {
8. a[i] = rand() % 57 + 65;
9. }
10. a[len] = '\0';
12. return a;
13. }
14. Функция print для вывода элементов списка в консоль.
15. void print(queue<char\*> qu, char\*\*& arr)
16. {
17. int n = qu.size();
18. for (int i = 0; i < n; i++)
19. {
20. arr[i] = qu.front();
21. cout << i + 1 << ".\t" << qu.front() << "\n";
22. qu.pop();
23. }
24. for (int i = 0; i < n; i++)
25. {
26. qu.push(arr[i]);
27. }
28. cout << endl;
29. }

**Блок-схема программы**

**Код программы**

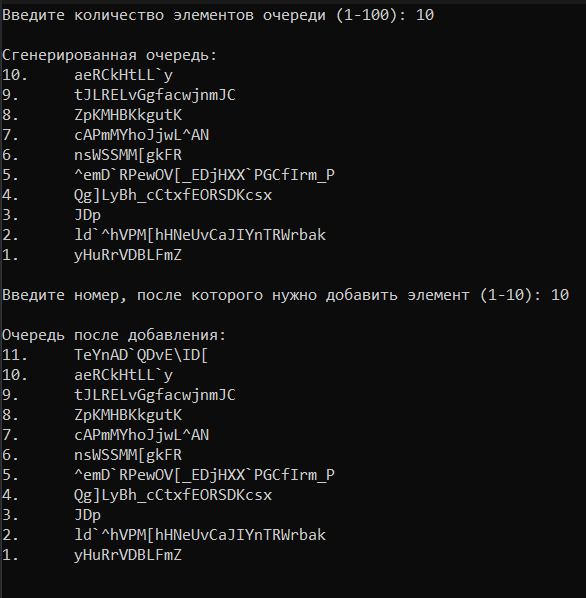
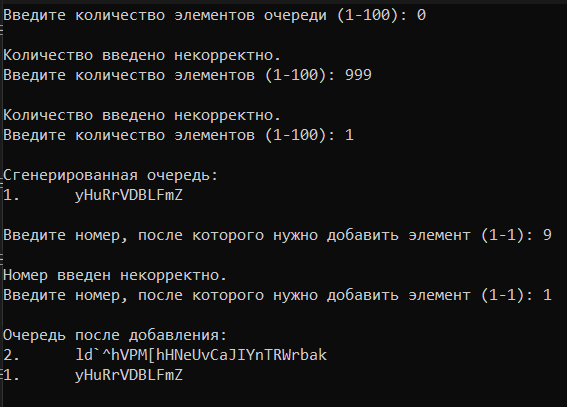
**Реализация через структуры**

1. #include <iostream>
2. #include <ctime>
3. using namespace std;
5. struct Queue
6. {
7. char\* data;
8. Queue\* next;
9. };
11. char\* generate\_string()
12. {
13. int len = rand() % 30 + 1;
14. char\* a = new char[len + 1];
15. for (int i = 0; i < len; i++)
16. {
17. a[i] = rand() % 57 + 65;
18. }
19. a[len] = '\0';
21. return a;
22. }
24. Queue\* make(int n, Queue\*& first, Queue\*& last)
25. {
26. if (n == 0) return NULL;
27. Queue\* p = new Queue;
28. p->data = generate\_string();
29. p->next = NULL;
30. first = p;
31. last = p;
32. for (int i = 1; i < n; i++)
33. {
34. Queue\* h = new Queue;
35. h->data = generate\_string();
36. h->next = p;
37. last = h;
38. p = last;
39. }
40. return first;
41. }
43. void print\_queue(Queue\* last, int size)
44. {
45. Queue\* p = last;
46. int k = size;
47. while (p != NULL)
48. {
49. int i = 0;
50. cout << k << ".\t";
51. while (p->data[i])
52. {
53. cout << p->data[i];
54. i++;
55. }
56. cout << endl;
57. k--;
58. p = p->next;
59. }
60. cout << endl;
61. }
63. char\* pop(Queue\*& first, Queue\*& last)
64. {
65. Queue\* p = last;
66. int k = 0;
67. while (p != NULL)
68. {
69. p = p->next;
70. k++;
71. }
72. p = last;
73. if (k == 1)
74. {
75. char\* str = last->data;
76. delete p;
77. last = NULL;
78. first = NULL;
79. return str;
80. }
81. else
82. {
83. while (p->next->next != NULL)
84. {
85. p = p->next;
86. }
87. Queue\* t = p;
88. char\* str = p->next->data;
89. p = p->next;
90. t->next = NULL;
91. first = t;
92. delete p;
93. return str;
94. }
95. }
97. Queue\* push(char\* n, Queue\*& last)
98. {
99. Queue\* p = new Queue;
100. p->data = n;
101. p->next = last;
102. last = p;
103. return last;
104. }
106. int main()
107. {
108. setlocale(LC\_ALL, "Russian");
109. int n, pos;
110. Queue\* first, \* last;
111. cout << "Введите количество элементов очереди (1-100): ";
112. cin >> n;
113. while (n < 1 || n > 99)
114. {
115. cout << "\nКоличество введено некорректно.\n";
116. cout << "Введите количество элементов (1-100): ";
117. cin >> n;
118. }
119. make(n, first, last);
120. cout << "\nСгенерированная очередь:\n";
121. print\_queue(last, n);
123. cout << "Введите номер, после которого нужно добавить элемент (1-" << n << "): ";
124. cin >> pos;
125. while (pos < 1 || pos > n)
126. {
127. cout << "\nНомер введен некорректно.\n";
128. cout << "Введите номер, после которого нужно добавить элемент (1-" << n << "): ";
129. cin >> pos;
130. }
131. char\*\* mas = new char\* [n];
133. for (int i = 0; i < n; i++)
134. {
135. mas[i] = pop(first, last);
136. }
137. for (int i = 0; i < pos; i++)
138. {
139. push(mas[i], last);
140. }
141. char\* tmp = generate\_string();
142. push(tmp, last);
143. for (int i = pos; i < n; i++)
144. {
145. push(mas[i], last);
146. }
147. n++;
148. cout << "\nОчередь после добавления:\n";
149. print\_queue(last, n);
151. return 0;
152. }

**Реализация через STL**

1. #include <iostream>
2. #include <ctime>
3. #include <queue>
4. using namespace std;
6. char\* generate\_string()
7. {
8. int len = rand() % 30 + 1;
9. char\* a = new char[len + 1];
10. for (int i = 0; i < len; i++)
11. {
12. a[i] = rand() % 57 + 65;
13. }
14. a[len] = '\0';
16. return a;
17. }
19. void print(queue<char\*> qu, char\*\*& arr)
20. {
21. int n = qu.size();
22. for (int i = 0; i < n; i++)
23. {
24. arr[i] = qu.front();
25. cout << i + 1 << ".\t" << qu.front() << "\n";
26. qu.pop();
27. }
28. for (int i = 0; i < n; i++)
29. {
30. qu.push(arr[i]);
31. }
32. cout << endl;
33. }
35. int main()
36. {
37. setlocale(LC\_ALL, "Russian");
38. int n, pos;
39. cout << "Введите количество элементов очереди (1-100): ";
40. cin >> n;
41. while (n < 1 || n > 99)
42. {
43. cout << "\nКоличество введено некорректно.\n";
44. cout << "Введите количество элементов (1-100): ";
45. cin >> n;
46. }
48. char\*\* arr = new char\* [n];
50. queue<char\*> qu;
51. for (int i = 0; i < n; i++)
52. {
53. qu.push(generate\_string());
54. }
55. print(qu, arr);
57. cout << "Введите номер, после которого нужно добавить элемент (1-" << n << "): ";
58. cin >> pos;
59. while (pos < 1 || pos > n)
60. {
61. cout << "\nНомер введен некорректно.\n";
62. cout << "Введите номер, после которого нужно добавить элемент (1-" << n << "): ";
63. cin >> pos;
64. }
65. for (int i = 0; i < n; i++)
66. {
67. arr[i] = qu.front();
68. qu.pop();
69. }
70. for (int i = 0; i < pos; i++)
71. {
72. qu.push(arr[i]);
73. }
74. qu.push(generate\_string());
76. for (int i = pos; i < n; i++)
77. {
78. qu.push(arr[i]);
79. }
80. cout << "\nОчередь после добавления: \n";
81. print(qu, arr);
82. n++;
83. for (int i = 0; i < n; i++)
84. {
85. qu.pop();
86. }
88. for (int i = 0; i < n; i++)
89. {
90. delete[] arr[i];
91. }
92. }

**Скриншоты результатов работы программы**

****