Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
Пермский национальный исследовательский политехнический университет
Электротехнический факультет
Кафедра информационных технологий и автоматизированных систем

Лабораторная работа №11

"Очереди"

Вариант: 12

Выполнил студент ИВТ-24-2б:
Шишкин Максим Григорьевич

_____
(дата, подпись)

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

_____
(дата, подпись)

Пермь 2025

Содержание

1 Постановка задачи..... 3

2 Код на c++..... 4-7

3 Результат работы программы 7

4 Блок-схема..... 7-11

Ссылка на github..... 11

1 Постановка задачи

Записи в очереди содержат ключевое поле типа *char(строка символов). Сформировать очередь. Удалить из него Элементы, с одинаковыми ключевыми полями. Добавить элемент после элемента с заданным ключевым полем.

2 Код на C++

```cpp
#include <iostream>
#include <string>
#include <clocale>

using namespace std;

const int MAX = 100; // Максимальный размер очереди

struct Queue {
    string items[MAX];
    int head;  // Индекс первого элемента
    int tail;   // Индекс последнего элемента
    int size;  // Количество элементов в очереди
};

// Инициализация очереди
void init(Queue& q) {
    q.head = 0;
    q.tail = -1;
    q.size = 0;
}

// Проверка на пустоту очереди
bool isEmpty(const Queue& q) {
    return q.size == 0;
}

// Проверка на заполненность очереди
bool isFull(const Queue& q) {
    return q.size == MAX;
}

// Добавление элемента в очередь
bool enqueue(Queue& q, const string& key) {
    if (isFull(q)) {
        cout << "Очередь переполнена!" << endl;
        return false;
    }
    q.tail++;
    q.items[q.tail] = key;
    q.size++;
    return true;
}

// Удаление элемента из очереди
string dequeue(Queue& q) {
    if (isEmpty(q)) {
```

```
48          cout << "Очередь пуста!" << endl;
49          return "";
50      }
51      string temp = q.items[q.head];
52
53      // Сдвигаем элементы к началу очереди
54      for (int i = 0; i < q.tail; ++i) {
55          q.items[i] = q.items[i + 1];
56      }
57
58      q.tail--;
59      q.size--;
60      return temp;
61  }
62
63  // Печать очереди
64  void printQueue(const Queue& q) {
65      if (isEmpty(q)) {
66          cout << "Очередь пуста!" << endl;
67          return;
68      }
69      cout << "Очередь: ";
70      for (int i = q.head; i <= q.tail; ++i) {
71          cout << q.items[i] << " ";
72      }
73      cout << endl;
74  }
75
76  // Добавление элемента после заданного
77  bool addAfter(Queue& q, const string& targetKey, const string& newKey) {
78      Queue tempQueue;
79      init(tempQueue);
80      bool found = false;
81      int originalSize = q.size; //Сохраняем размер, так как dequeue его меняет
82
83      for (int i = 0; i < originalSize; ++i) {
84          string item = dequeue(q);
85          enqueue(tempQueue, item);
86          if (item == targetKey && !found) {
87              enqueue(tempQueue, newKey);
88              found = true;
89          }
90      }
91
92      // Возвращаем элементы обратно в исходную очередь
93      while (!isEmpty(tempQueue)) {
94          enqueue(q, dequeue(tempQueue));
```

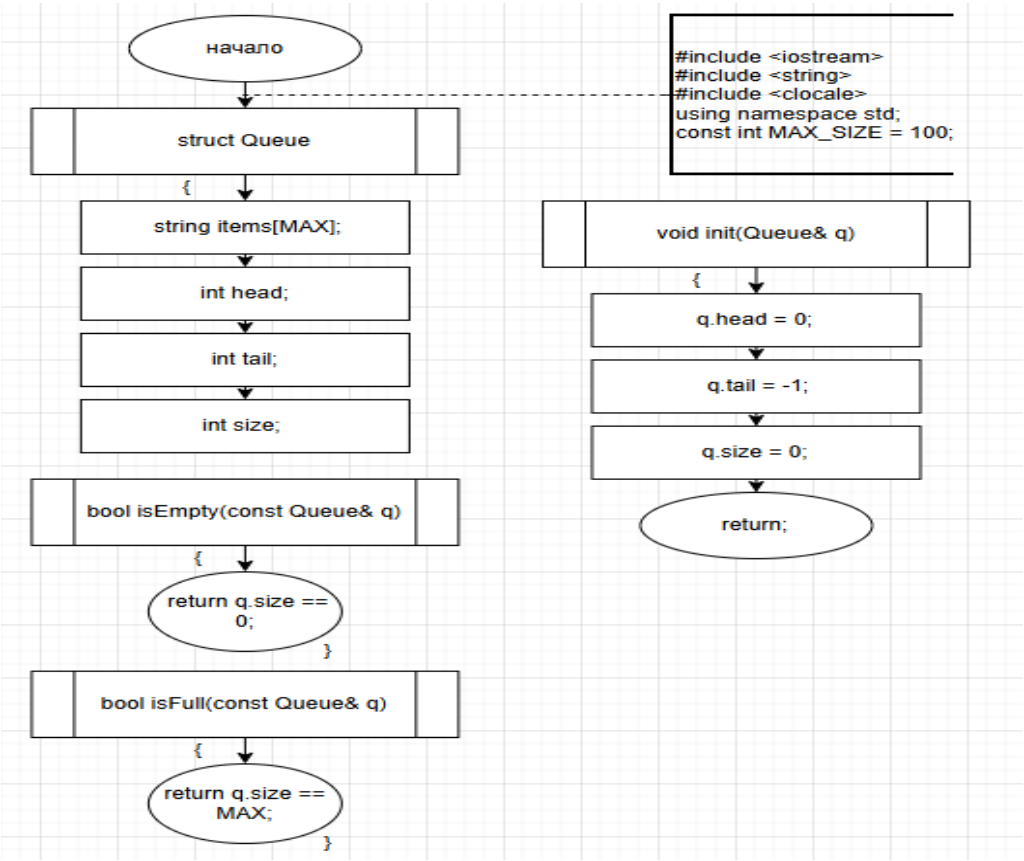
```
95          }
96
97          if (!found) {
98              cout << "Элемент с ключом '" << targetKey << "' не найден." << endl;
99          }
100         return found;
101     }
102
103     // Удаление дубликатов
104     void removeDuplicates(Queue& q) {
105         Queue tempQueue;
106         init(tempQueue);
107         string uniqueItems[MAX];
108         int uniqueCount = 0;
109         int originalSize = q.size; //Сохраняем размер, так как dequeue его меняет
110
111
112         for (int i = 0; i < originalSize; ++i) {
113             string item = dequeue(q);
114             bool duplicate = false;
115             for (int j = 0; j < uniqueCount; ++j) {
116                 if (uniqueItems[j] == item) {
117                     duplicate = true;
118                 }
119             }
120             if (!duplicate) {
121                 uniqueItems[uniqueCount] = item;
122                 uniqueCount++;
123                 enqueue(tempQueue, item);
124             }
125         }
126
127         // Возвращаем уникальные элементы обратно в исходную очередь
128         while (!isEmpty(tempQueue)) {
129             enqueue(q, dequeue(tempQueue));
130         }
131     }
132
133     int main() {
134         setlocale(LC_ALL, "ru");
135         string strArr[] = { "sos", "acer", "dog", "sos", "date", "acer", "cat" };
136         int sz = sizeof(strArr) / sizeof(strArr[0]);
137
138         Queue myQueue;
139         init(myQueue);
140
141         cout << "Исходный массив: ";
```

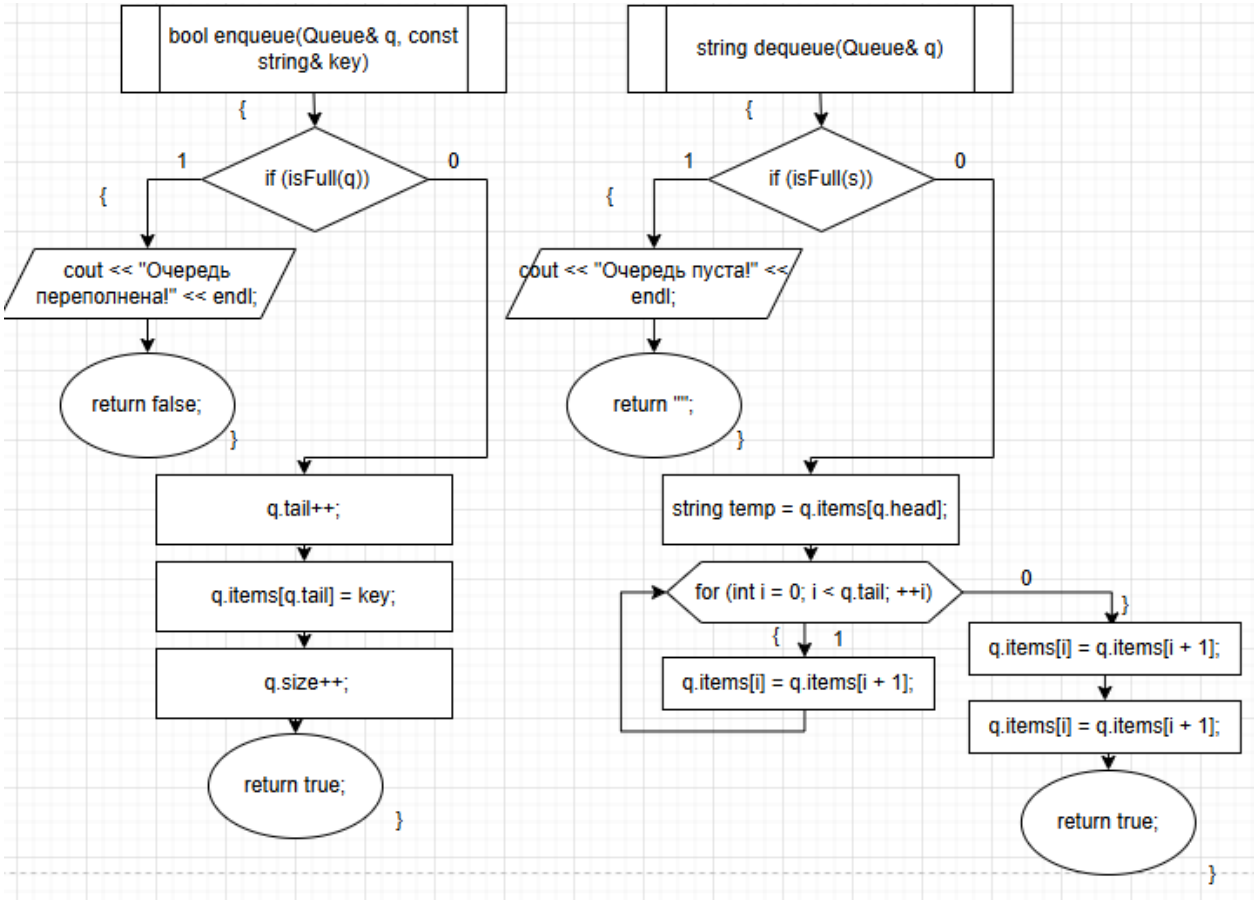
```
142     for (int i = 0; i < sz; ++i) {
143         cout << strArr[i] << " ";
144     }
145     cout << endl;
146
147     for (int i = 0; i < sz; ++i) {
148         enqueue(myQueue, strArr[i]);
149     }
150
151     cout << "Очередь: ";
152     printQueue(myQueue);
153
154     removeDuplicates(myQueue);
155     cout << "Очередь после удаления дубликатов: ";
156     printQueue(myQueue);
157
158     addAfter(myQueue, "dog", "mouse");
159     cout << "Очередь после добавления элемента: ";
160     printQueue(myQueue);
161
162     return 0;
163 }
```

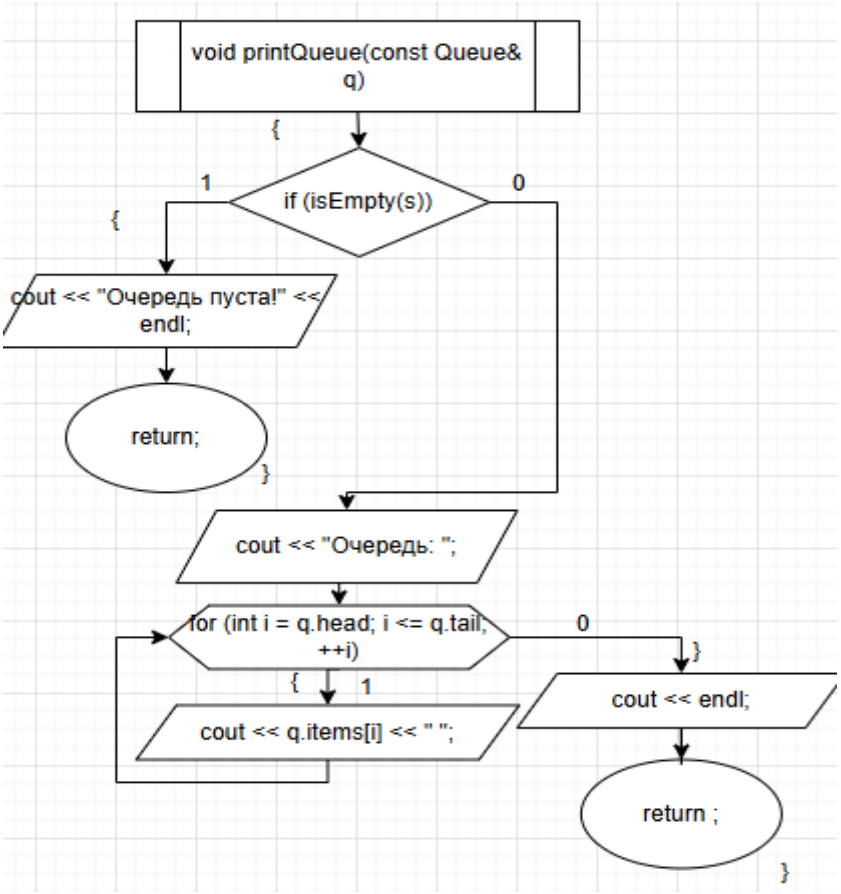
3 Результат работы программы

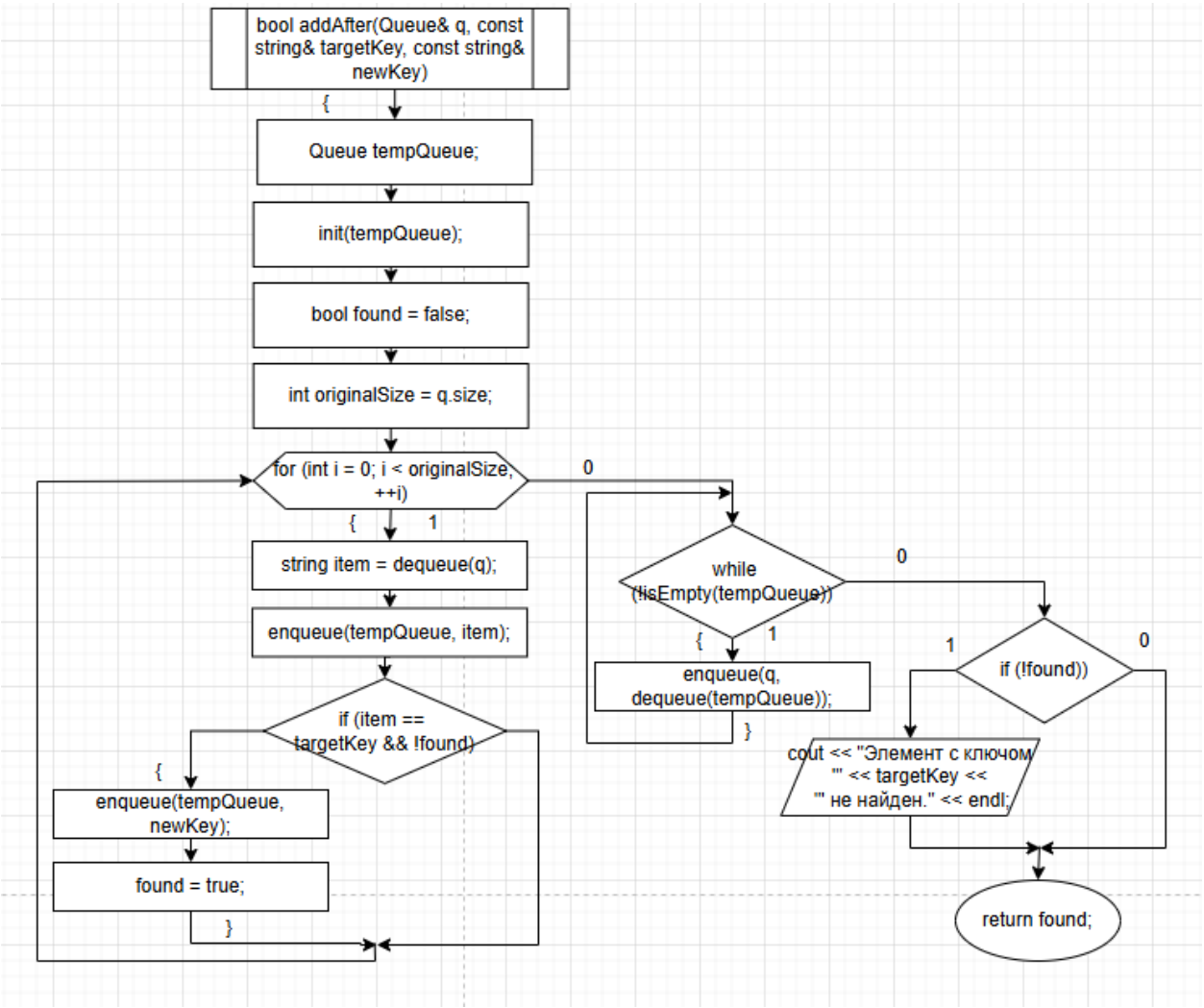
```
Исходный массив: sos acer dog sos date acer cat
Очередь: Очередь: sos acer dog sos date acer cat
Очередь после удаления дубликатов: Очередь: sos acer dog date cat
Очередь после добавления элемента: Очередь: sos acer dog mouse date cat
```

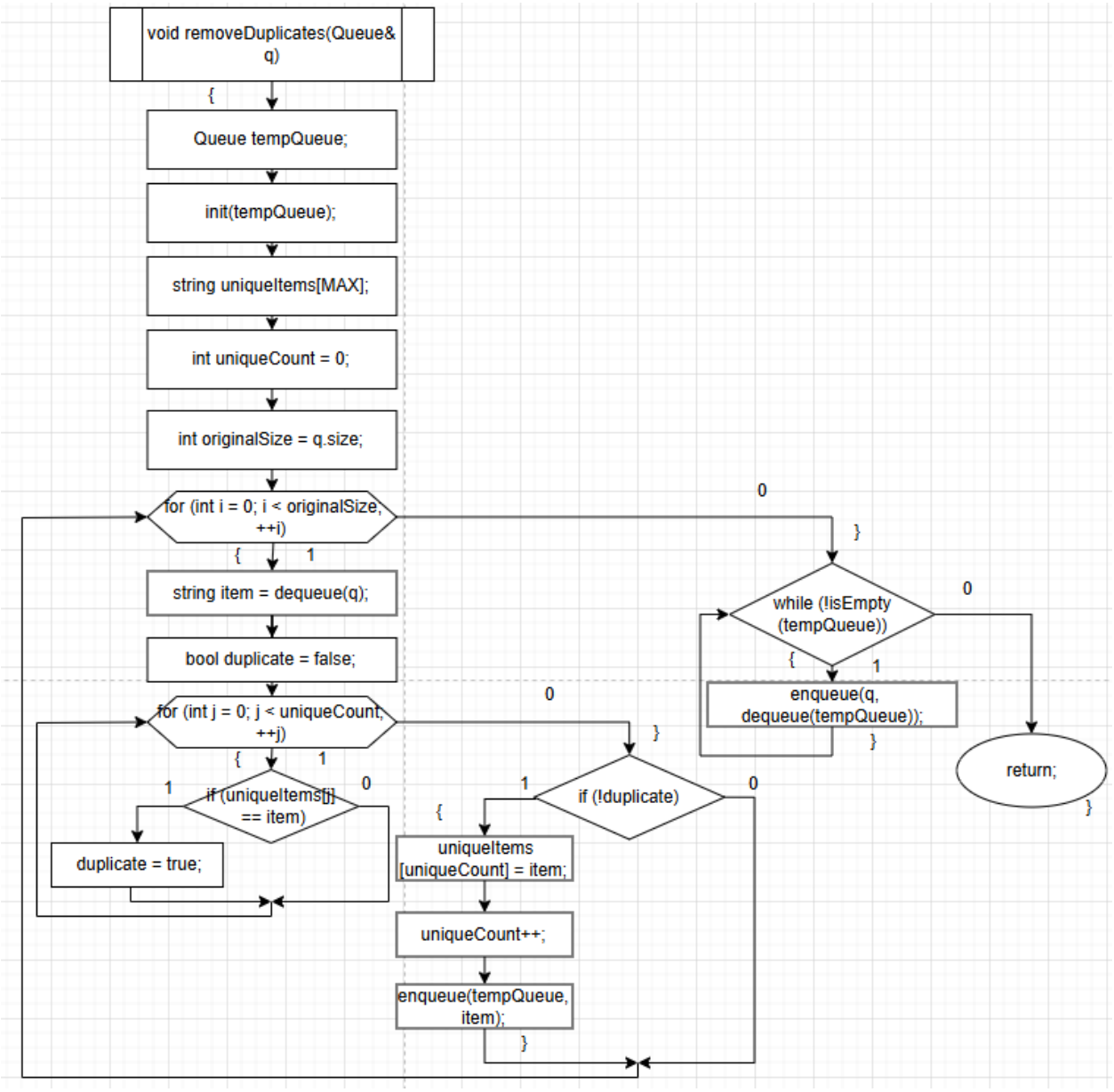
4 Блок-схема

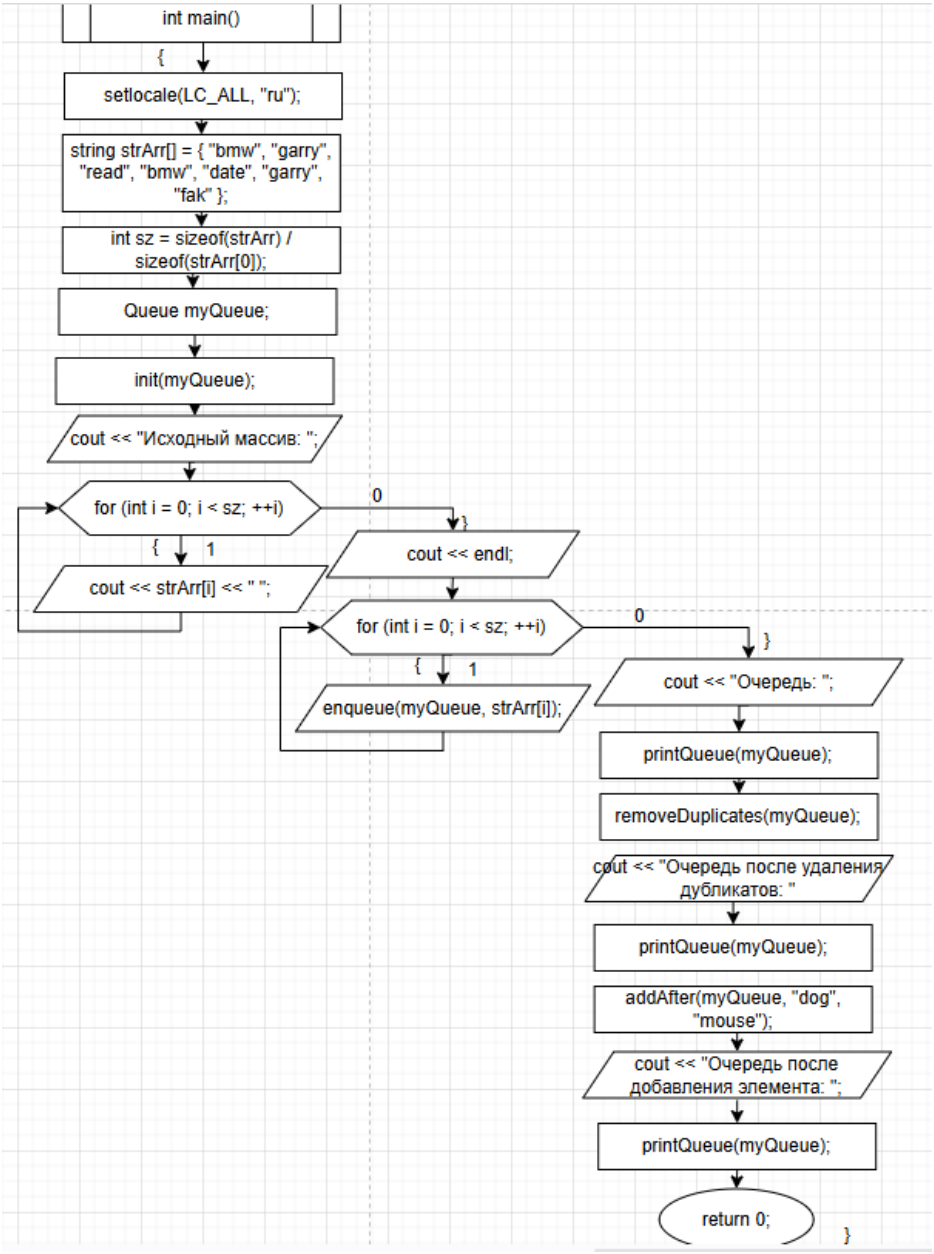



bool enqueue(Queue& q, const string& key)
{
if (isFull(q))
1
0
{
cout << "Очередь переполнена!" << endl;
return false;
}
q.tail++;
q.items[q.tail] = key;
q.size++;
return true;
}
string dequeue(Queue& q)
{
if (isFull(s))
1
0
{
cout << "Очередь пуста!" << endl;
return "";
}
string temp = q.items[q.head];
for (int i = 0; i < q.tail; ++i)
0
}
{
1
q.items[i] = q.items[i + 1];
q.items[i] = q.items[i + 1];
q.items[i] = q.items[i + 1];
return true;
}


void printQueue(const Queue& q)
{
if (isEmpty(s))
1
0
{
cout << "Очередь пуста!" << endl;
return;
}
cout << "Очередь: ";
for (int i = q.head; i <= q.tail; ++i)
0
}
{
1
cout << q.items[i] << " ";
cout << endl;
return ;
}


bool addAfter(Queue& q, const string& targetKey, const string& newKey)
{
Queue tempQueue;
init(tempQueue);
bool found = false;
int originalSize = q.size;
for (int i = 0; i < originalSize; ++i)
0
{
1
string item = dequeue(q);
enqueue(tempQueue, item);
if (item == targetKey && !found)
{
enqueue(tempQueue, newKey);
found = true;
}
while (!isEmpty(tempQueue))
0
{
1
enqueue(q, dequeue(tempQueue));
}
if (!found))
1
0
cout << "Элемент с ключом "" << targetKey << "" не найден." << endl;
return found;


void removeDuplicates(Queue& q)
{
Queue tempQueue;
init(tempQueue);
string uniqueItems[MAX];
int uniqueCount = 0;
int originalSize = q.size;
for (int i = 0; i < originalSize; ++i)
{ 1
string item = dequeue(q);
bool duplicate = false;
for (int j = 0; j < uniqueCount; ++j)
{ 1
1 if (uniqueItems[j] == item) 0
duplicate = true;
0
}
if (!duplicate)
1 {
uniqueItems [uniqueCount] = item;
uniqueCount++;
enqueue(tempQueue, item);
}
0
}
while (!isEmpty (tempQueue))
{ 1
enqueue(q, dequeue(tempQueue));
}
0
return;
}



5 Ссылка на github

ссылка на github - https://github.com/MAKSPOWERO/mas1