Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №11**

Дисциплина: основы алгоритмизации и программирования

Тема: “Динамические структуры данных” в C++

Выполнил работу

студент группы ИВТ-20-1б

Матяж Владимир Олегович

Проверил

Доцент кафедры ИТАС

Полякова Ольга Андреевна

Пермь 2020

**Цель задачи**

Знакомство с динамическими информационными структурами на примере очереди.

Постановка задачи

Написать программу, в которой создаются динамические структуры и выполнить их обработку в соответствии со своим вариантом.

Записи в очереди содержат ключевое поле типа char (строка символов). Сформировать очередь. Удалить элемент с заданным ключом. Добавить К элементов перед элементом с заданным номером.

Для выполнения данной работы использован язык программирования C++ в программной среде Visual Studio 2019.

Анализ задачи, реализованной с помощью STL - библиотеки

Определить какие операции должны быть выполнены по заданию:

* Формирование очереди;
* Вывод очереди;
* Удаление элемента с заданным ключом;
* Добавление элементов после элемента с заданным номером;
* Создание массива типа char;
* Важный момент:

Для использования функций STL - библиотеки требуется подключить библиотеку очереди **<queue>;**

* Все операции реализуем по принципу очереди FIFO ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) first in — first out, «первым пришёл — первым вышел»).

Формирование очереди:

Создание пустой очереди:

queue<char> qu;

Просим пользователя ввести кол-во элементов. Данные в структуру введены через цикл for и функции push() – добавление элемента в конец очереди.

Вывод очереди:

Вывод производится с помощью отдельной функции, в качестве параметра принимает очередь:

void print(queue<char> qu) {

int p = qu.size();

for (int i = 0; i < p; i++)

{

arr[i] = qu.front();

cout << qu.front() << " ";

qu.pop();

}

for (int i = 0; i < p; i++)

qu.push(arr[i]);

cout << endl;

}

Сохраняем размер очереди в переменную типа int с помощью функции size(). Выводим очередь с помощью цикла for. Поскольку нужно вывести очередь, затем выполнить операции над ней, сохраняем значения первого элемента в массив с помощью функции front(), выводим значение на консоль, удаляем первый элемент с помощью функции pop(), повторяем со всеми элементами. Затем переносим элементы массива в очередь с помощью цикла for и функции добавления элемента в конец очереди push().

Удаляются элементы очереди с заданным ключом.

Для реализации удаления элементов очереди используем цикл while. В нем происходит проверка элемента с помощью условного оператора if на равенство ключу, если равен то удаляем его с помощью функции pop() и уменьшаем кол-во элементов на 1, иначе переносим в массив. Переносим элементы массива в очередь. Выполняем вывод очереди после удаления, вызовом функции из пункта 3.

Добавление элементов:

Просим ввести пользователя номер элемента, после которого нужно добавить элементы и их кол-во, для этого понадобится две целочисленных переменные. С помощью цикла for и функции front() – доступ к первому элементу переносим элементы в массив, переносим элементы из массива в очередь до элемента с заданным номером включительно. С помощью цикла for и функции push() добавляем элементы в очередь, аналогично переносим остальные элементы массива в очередь.

Выводим очередь на консоль, вызовом функции из пункта 3.

Анализ задачи, реализованной через структуру.

Определить какие операции должны быть выполнены по заданию:

* Создание структуры;
* Формирование очереди;
* Вывод очереди;
* Удаление элемента с заданным ключом;
* Добавление элементов после элемента с заданным номером;
* Создание массива типа char;
* Все операции реализуем по принципу очереди FIFO ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) first in — first out, «первым пришёл — первым вышел»).

Разработка структуры. Структура состоит из ключа (значение, которое будет хранить элемент очереди) и указатель на следующий элемент очереди.

struct Queue {

char data;

Queue\* next;

};

Создаем указатели на первый и последний элемент. Главная функция будет аналогична, главной функции задачи, реализованной через STL.

Для формирования очереди создадим функцию, в качестве параметров она будет принимать кол-во элементов, указатели на первый и последние элементы. В начале функции нужно сделать проверку, не равно ли кол-во элементов 0. Далее выделяем память под 1 элемент, присваиваем первый элемент, указателю на следующий элемент присваиваем NULL, т.к. он не на что не указывает. Указателям на первый и последний элемент присваиваем адрес первого элемента, т.к. он первый и последний. Далее с помощью цикла for выделяем память для нового элемента, присваиваем новый элемент, указываем на следующий элемент, указателю нового элемента присваивается указатель на последний элемент в очереди, указателю на последний элемент присваивается указатель на новый элемент.

Для печати очереди создадим функцию, создаем указатель и присваиваем адрес последнего элемента, с помощью цикла while выводим элементы очереди, пока не дойдет до первого, далее указатель будет равен NULL и цикл закончится.

Удаление элементов также производится с помощью специальной функции, параметром передаём указатель на первый и последний элементы очереди. Объявляем указатель p, чтобы сохранить последний элемент. Далее подсчёт количества элементов в очереди. Если в очереди находится один элемент, то возвращается значение этого элемента и удаляется из памяти, иначе осуществляется переход к предпоследнему элементу и его указатель сохраняется в переменную t. Переход к последнему элементу, сохраняется его значение, чтобы потом возвратить, и элемент удаляется. Указателю предпоследнего элемента присваивается NULL, а указателю на первый элемент очереди присваивается указатель предпоследнего элемента.

Создание функции добавления. В качестве параметра передаётся указатель на последний элемент очереди и значение элемента, который необходимо добавить. Выделяется память для нового элемента, присваивается значение, переданное параметром, указателю нового элемента присваивается указатель на последний элемент очереди, а указателю на последний элемент присваивается указатель на новый элемент.

Код реализованный с помощью STL - библиотеки

#include <iostream>

#include <queue> // Заголовок очереди

using namespace std;

int n, k, s;

char\* arr;

char d;

void print(queue<char> qu) {

int p = qu.size(); /\* Сохранили размер очереди

в переменную \*/

for (int i = 0; i < p; i++)

{ // Печать элементов очереди

arr[i] = qu.front();

cout << qu.front() << " ";

qu.pop();

}

for (int i = 0; i < p; i++) // Перенос элементов в массив

qu.push(arr[i]);

cout << endl;

}

int main() {

setlocale(LC\_ALL, "Rus");

arr = new char[n];

cout << "Введите размер" << endl;

cin >> n;

cout << "Введите ключ" << endl;

cin >> d;

queue<char> qu; // Создали пустую очередь

cout << "Введите " << n <<" элементов" << endl;

for (int i = 0; i < n; i++)

{

char a;

cin >> a;

qu.push(a); // Добавление элементов в очередь

}

print(qu); //Печать очереди

int j = 0;

while (j < n) { // Удаление элемента с заданным ключем

if (d == qu.front())

{

qu.pop(); // Удаление элемента, равного ключу

n--;

}

else

{

arr[j] = qu.front(); // Перенос элемента в массив

qu.pop(); // Удаление элемента из очереди

j++;

}

}

for (int i = 0; i < n; i++) // Перенос элементов в массив

qu.push(arr[i]);

cout << "Очередь после удаления" << endl;

print(qu); // Печать очереди

cout << "Введите номер элемента, полсе которго нужно добавление" << endl;

cin >> s;

cout << "Введите количество элементов для добавления" << endl;

cin >> k;

if (s >= n)

{

cout << "Введите другое значение s, меньшее " << n << endl;

cin >> s;

}

for (int i = 0; i < n; i++)

{

arr[i] = qu.front(); // Перенос элемента в массив

qu.pop(); // Удаление элемента

}

for (int i = 0; i <= s; i++)

{

qu.push(arr[i]); // Перенос элементов до элемента с заданным номером из массива

}

cout << "Введите " << k << " элементов" << endl;

for (int i = 1; i <= k; i++) // Добавление K элементов после элемента с заданным номером

{

char a;

cin >> a;

qu.push(a);

}

for (int i = s + 1; i < n; i++)

{

qu.push(arr[i]); // Перенос остальных элементов из массива

}

cout << "Очередь после добавления " << endl;

print(qu);

for (int i = 0; i < n+k; i++)

{

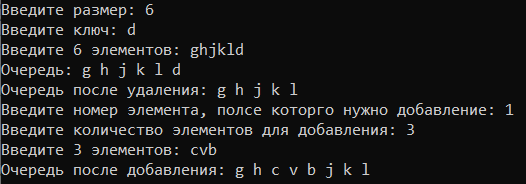
qu.pop();

}

return 0;

}

Результат работы



Код реализованный с помощью структуры

#include <iostream>

using namespace std;

int n;

char key;

struct Queue { // Создаем структуру

char data;

Queue\* next;

};

Queue\* make\_queue(int n, Queue\*& first, Queue\*& last) {

if (n == 0) return NULL;

Queue\* p = new Queue; // Выделение памяти под 1 элемент

char a;

cin >> a;

p->data = a;

p->next = NULL; /\* Так как он первый, то он ни на что не указывает \*/

first = p; // Новый элемент является первым

last = p; // И последним

for (int i = 2; i <= n; i++) {

Queue\* h = new Queue; // Выделение памяти для нового элемента

cin >> a;

h->data = a;

h->next = p; // Указываем на следующий элемент

last = h; /\* Показываем, что введённый элемент стал последним \*/

p = last; // Перешли к последнему элементу

}

return first;

}

void print\_queue(Queue\* last)

{ /\* Печать очереди без удаления элементов \*/

Queue\* p = last; // p указывает на последний элемент

while (p != NULL)

{ // Печать с конца очереди

cout << p->data << " "; // Печать

p = p->next; // Переход к следующему элементу

}

cout << endl;

}

char pop(Queue\*& first, Queue\*& last) {

// Функция удаления элемента

Queue\* p = last;

int k = 0; // Кол-во элементов в очереди

while (p != NULL) { // Считаем кол-во элементов

k++;

p = p->next;

}

p = last;

if (k == 1) { /\* Если элемент один, то обнуляем указатели и возвращаем последний элемент \*/

char t = last->data;

delete p;

last = NULL;

first = NULL;

return t;

}

else { // Если больше 1

while (p->next->next != NULL) // Переходим к предпоследнему элементу

{

p = p->next;

}

Queue\* t = p; // Указатель на предпоследний элемент

char e = p->next->data; // Сохранение значения последнего элемента

p = p->next; // Переход к последнему элементу

t->next = NULL; // Предпоследний указатель указывает на NULL

first = t; // И он теперь первый элемент

delete p; // Удаление последнего элемента

return e; // Возвращение первого элемента

}

}

Queue\* push(char n, Queue\*& last) {

Queue\* p = new Queue; // Выделение памяти для нового элемента

p->data = n; // Присвоение

p->next = last; // Наш новый элемент указывает на следующий

last = p; // Новый элемент стал последним

return last;

}

int main() {

setlocale(LC\_ALL, "Rus");

cout << "Введите кол-во элементов " << endl;

cin >> n;

cout << "Введите ключ для удаления" << endl;

cin >> key;

Queue\* first, \* last; /\* Указатели на первый и последний

элементы

очереди \*/

make\_queue(n, first, last); // Формирование очереди

print\_queue(last); // Печать очереди

char \*mas = new char[n];

int i = 0;

while(i < n)

{

char t = pop(first, last); // Удаляем элемент, сохраняя в переменную

if (t != key) // Проверяем на равенство элементу для удаления

{

mas[i] = t;

i++;

}

else

{

n--;

}

}

for (int i = 0; i < n; i++) // Перенос элементов в массив

{

push(mas[i], last);

}

cout << "Очередь после удаления: ";

print\_queue(last); // Печать очереди

cout << "Номер элемента для добавления " << endl;

int number;

cin >> number;

cout << "Кол-во элементов для добавления " << endl;

int kol;

cin >> kol;

for (int i = 0; i < n; i++)

// Перенос элементов в массив

mas[i] = pop(first, last);

for (int i = 0; i <= number; i++)

// Добавление первой половины из массива

push(mas[i], last);

cout << "Введите " << kol << " элементов" << endl;

for (int i = 0; i < kol; i++) // Добавление новых элементов

{

char a;

cin >> a;

push(a, last);

}

for (int i = number + 1; i < n; i++) /\* Добавление второй половины из массива \*/

push(mas[i], last);

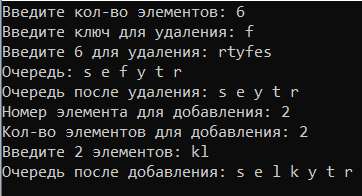
cout << "Очередь после добавления: ";

print\_queue(last); // Печать очереди

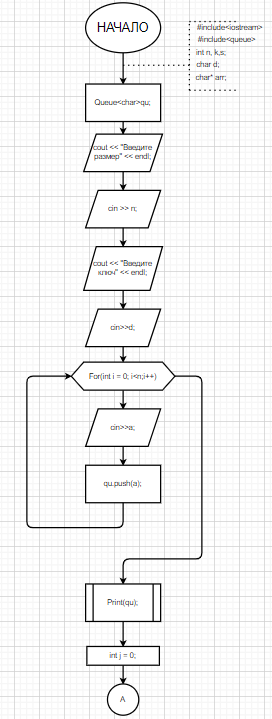
return 0;

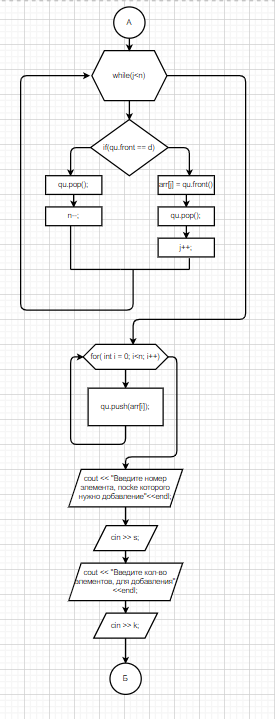
}

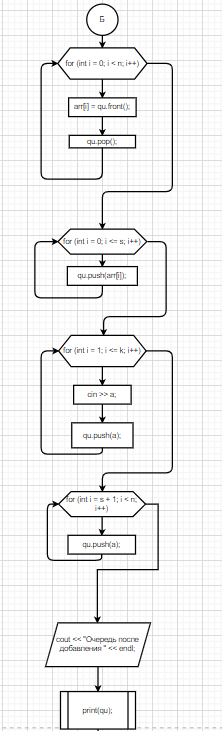
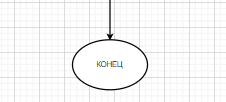
## Результат работы

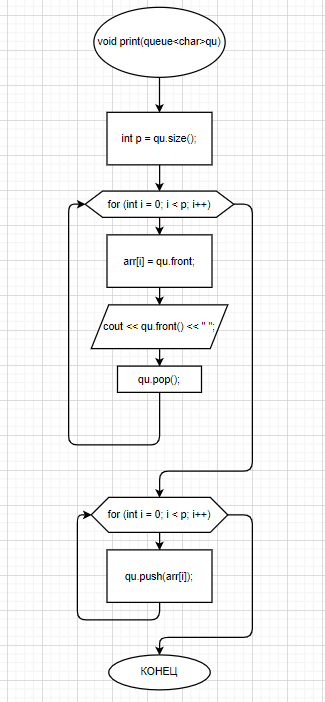


Блок-схема через STL









Блок-схема реализации через структуру

