Лабораторная работа №12
Тема: «Очереди»

Цель работы: сформировать знания и умения по работе с подпрограммами, приобрести навыки написания программ с использованием очередей.

Время выполнения: 4 часа.

Теоретические сведения

Очередь — это структура данных (как было сказано выше), которая построена по принципу FILO (last in — last out: первым пришел — последним вышел). В C++ уже есть готовый STL контейнер — queue.

В очереди, если вы добавите элемент, который вошел первым, то он выйдет тоже самым первым. Получается, если вы добавите 4 элемента, то первый добавленный элемент выйдет первым.

Чтобы понять принцип работы очереди вы можете представить себе магазинную очередь. И вы стоите посреди нее, чтобы вы оказались напротив кассы, сначала понадобится всех впереди стоящих людей обслужить. А вот для последнего человека в очереди нужно, чтобы кассир обслужил всех людей кроме него самого.

Если вы хотите использовать шаблон очереди в C++, то вам сначала нужно подключить библиотеку — <queue>.

Дальше для объявления очереди нужно воспользоваться конструкцией ниже.

queue <тип данных> <имя>;

Сначала нам нужно написать слова queue.

Дальше в <тип данных> мы должны указать тот тип, которым будем заполнять нашу очередь.

И в конце нам остается только указать название очереди.

Вот пример правильного объявления:

queue <int> q;

Методы очереди

Метод — это та же самая функция, но она работает только с контейнерами STL. Например, очередь и стек.

Для работы с очередью вам понадобится знать функции: push(), pop(), front(), back(), empty(). Кстати, если хотите узнать, как в C++ работают

функции и как их правильно использовать в проекте, то можете узнать все это здесь.

Для добавления в очередь нового элемента нужно воспользоваться функцией — push(). В круглых скобках должно находится значение, которое мы хотим добавить.

Если нам понадобилось удалить первый элемент нужно оперировать функцией pop(). В круглых скобках уже не чего не нужно указывать, но по правилам они в обязательном порядке должны присутствовать! Эти функции тоже не нуждаются в указании аргумента: empty(), back() и front().

Если вам понадобилось обратиться к первому элементу очереди, то вам понадобится функция front().

Чтобы обратиться к последнему элементу в очереди вам поможет функция back().

Чтобы узнать пуста ли очередь нужно воспользоваться функцией empty().

Если ваша очередь пуста — возвратит true.

Если же в ней что-то есть — возвратит false.

Ниже мы использовали все выше перечисленные методы:

```
#include <iostream>
#include <queue>  // подключили библиотеку queue

using namespace std;

int main() {
  setlocale(LC_ALL,"rus");
  queue <int> q;  // создали очередь q

  cout << "Пользователь, пожалуйста введите 7 чисел: " << endl;

  for (int h = 0; h < 7; h++) {
    int a;

    cin >> a;

    q.push(a);  // добавляем в очередь элементы
  }

  cout << endl;
```

```
    cout << "Первый элемент в очереди: " << q.front() << endl;  // выводим первый
                                                                // элемент очереди
    q.pop();  // удаляем элемент из очереди

    cout << "Новый первый элемент (после удаления): " << q.front() << endl;

    if (!q.empty()) cout << "Очередь не пуста!";  // проверяем пуста ли очередь (нет)

    system("pause");
    return 0;
}
```

Создание очереди с помощью массива

Как мы говорили выше, очередь можно реализовать через массив. Обычно, если кто-то создает такую очередь, то массив называют queue. Мы бы также назвали массив, но это уже зарезервированное слова в C++. Поэтому его назовем так, как назвали выше шаблон очереди — q.

Для реализации нам понадобится создать две дополнительные переменные start и ends. start будет указывать на первый элемент очереди, а ends на последний элемент.

Чтобы обратится к последнему элементу нам придется использовать эту конструкцию — queue[ends]. Обращение к первому элементу будет выглядеть аналогично queue[start].

Если понадобится удалить элемент из очереди, то придется всего лишь уменьшить переменную start на один.

«А как же проверить пуста ли очередь?» — спросите вы. Для этого мы просто проверим условие start == ends:

Если результатом условия будет true, то очередь пуста.
Если же результат условия false, значит очередь чем-то заполнена.
Ниже находится живой пример создания такой очереди:

```
setlocale(LC_ALL,"rus");
int q[7];  // создали массив q
```

```
int start = 0, ends = 0;  // создали переменные начала и конца очереди

cout << "Пользователь, пожалуйста введите 7 чисел: " << endl;

for (int h = 0; h < 7; h++) { int a; cin >> a;
  int a;

  cin >> a;

  q[ends++] = a;  // добавляем элементы в очередь(массив)
 }

cout << "Первый элемент в очереди: " << q[start] << endl;

start++;  // удаляем первый элемент(увеличиваем индекс начала очереди на 1)

cout << "Новый первый элемент (после удаления): " << q[start] << endl;

cout << "Последний элемент в очереди: " << q[ends - 1];  // выводим последний
                                                         // элемент очереди
if (start != ends) cout << "Очередь заполнена!";         // проверяем пуста ли очередь
```

Очередь с приоритетом

Очередь с приоритетом (priority_queue) — это обычная очередь, но в ней новый элемент добавляется в то место, чтобы очередь была отсортирована по убыванию.

Так самый большой элемент в приоритетной очереди будет стоять на первом месте.

Для объявления шаблона очереди с приоритетом нужно использовать конструкцию ниже:

priority_queue <тип данных> <имя>;

В начале нужно написать priority_queue.

Потом в скобках указать тип данных, который будет находится в очереди.

И конечно же в самом конце мы должны дать ей имя.

Для добавления элемента в очередь с приоритетом мы должны использовать функцию push(). Но чтобы обратится к первому элементу должны использоваться именно функция — top() (как и в стеке). А не функция — front().

Также нельзя использовать функцию back() для обращения к последнему элементу. Для приоритетной очереди она также не работает, как функция front().

Вот пример использования очереди с приоритетом в программе:

```
setlocale(LC_ALL,"rus");

priority_queue <int> priority_q;  // объявляем приоритетную очередь

cout << "Введите 7 чисел: " << endl;

for (int j = 0; j < 7; j++) { int a; cin >> a;

  priority_q.push(a);  // добавляем элементы в очередь
}
                                          // выводим первый
cout << "Первый элемент очереди: " << priority_q.top();  // элемент
```

Индивидуальные задания к лабораторной работе №12

1. Постройте очередь из 7-ти символов - ‘k’, ‘l’,‘m’, ‘n’, ‘o’, ‘p’, ‘r’. Выведите из очереди три символа ‘l’, ‘m’, ‘r’ и добавьте в конец очереди символ ‘s’. Результаты как промежуточных, так и конечных результатов отобразить на экране.
2. Разработайте программу, с помощью которой можно определить наибольший допустимый размер очереди с вещественным информационным полем. Найдите этот размер (число элементов в очереди).
3. Опишите очередь с вещественным информационным полем, и заполните ее элементами с клавиатуры. Выполните циклический сдвиг элементов в очереди так, чтобы в ее начале был расположен наибольший элемент.
4. Система состоит из трех процессоров P1, P2, P3, очереди F, стека S и распределителя задач R. В систему поступают запросы на выполнение задач трёх типов – T1, T2 и T3, каждая для своего процессора. Поступающие запросы ставятся в очередь. Если в начале очереди находится задача Ti и

процессор Pi свободен, то распределитель R ставит задачу на выполнение в процессор Pi, а если процессор Pi занят, то распределитель R отправляет задачу в стек и из очереди извлекается следующая задача. Если в вершине стека находится задача, процессор которой в данный момент свободен, то эта задача извлекается и отправляется на выполнение.

5. Дан набор из 10 чисел. Создать две очереди: первая должна содержать числа из исходного набора с нечетными номерами (1, 3, ..., 9), а вторая — с четными (2, 4, ..., 10); порядок чисел в каждой очереди должен совпадать с порядком чисел в исходном наборе. Вывести указатели на начало и конец каждой из полученных очередей.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретическую часть лабораторной работы.

2. Реализовать индивидуальное задание по вариантам, представленные в теоретических сведениях, сделать скриншоты работающих программ. Написать комментарии.

3. Написать отчет, содержащий:

1. Титульный лист, на котором указывается:

а) полное наименование министерства образование и название учебного заведения;

б) название дисциплины;

в) номер практического занятия;

г) фамилия преподавателя, ведущего занятие;

д) фамилия, имя и номер группы студента;

е) год выполнения лабораторной работы.

2. Индивидуальное задание из раздела «Теоретические сведения» с кодом, комментариями и скриншотами работающих программ.

3. Построение блок-схем.