Министерство образования и науки Российской Федерации

Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский**

**политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №11.2**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: Информационные и динамические структуры. Очереди

Вариант 18

Выполнил:

Студент группы РИС-20-1б

Тараканов Д. М.

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

Пермь

2021 год

# **Цель работы**

1) Получить практические навыки работы с очередями;

2) Получить практические навыки работы с очередями через STL;

# **Постановка задачи**

Задача состоит в разработке программы, в которой создаётся очередь и очередь через STL, выполнить обработку очередей в соответствии со своим вариантом:

(18 вариант)

Очередь

1. Сформировать очередь с информационным полем типа char.
2. Добавить в очередь элемент с заданным номером.

Очередь через STL

1. Сформировать очередь через STL с информационным полем типа char.
2. Добавить в очередь элемент с заданным номером.

# **Анализ задачи**

Очередь

1. Определить какие предстоит выполнить действия:

* Создать структура очередь в виде однонаправленного списка

struct Queue {

char data;

Queue\* next;

};

* Разработать функцию FillQueue для заполнения очереди

void FillQueue(int N, Queue \*&first) {

Queue\* p = new Queue;

char el;

cout << "Введите 1 элемент: ";

cin >> el;

p->data = el;

first = p;

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

Queue\* q = new Queue;

cout << "Введите " << i << " элемент: ";

cin >> el;

q->data = el;

p->next = q;

p = p->next;

}

p->next = nullptr;

};

* Разработать функцию AddElement для добавления в очередь элемент с заданным номером.

void AddElement(int size, Queue \*&first) {

int N = -1,count=0;

char key;

Queue\* p = first;

while ((N > size) || (N < 1)) {

if ((N > size) || (N < 1)) {

cout << "Введите номер элемента: ";

}

cin >> N;

}

cout << "Введите элемента который хотите добавить: ";

cin >> key;

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

p = p->next;

}

if (N != 1) {

Queue\* q = new Queue;

q->data = key;

q->next = p->next;

p->next = q;

q = q->next;

}

else {

Queue\* q = new Queue;

q->data = key;

q->next = first;

first = q;

}

};

2. С какими типами данных действие надо сделать, в каком виде эти данные будут представлены:

* Для запоминания длины очереди используем переменную size типа int.

int size = -1;

* Для данных в очереди используем переменную data типа char.

char data;

* Для хранения адреса следующего элемента используем указатель next типа Queue.

Queue\* next;

* Для хранения нового элемента используем переменную el типа char.

char el;

* Для обращения к очереди используем указатель first на первый элемент очереди типа Queue.

Queue\* first;

* Для функции FillQueue используем следующие элементы.

1. Длину очереди типа int.
2. Указатель на первый элемент очереди типа Queue

Сама функция имеет тип void так, как входе её работы функция не возвращает значений.

void FillQueue(int N, Queue \*&first)

* Для функции PrintQueue используем следующие элементы.

1. Указатель на первый элемент очереди типа Queue.

Сама функция имеет тип void так, как входе её работы функция не возвращает значений.

void PrintQueue(Queue\* first)

* Для функции AddElement используем следующие элементы.

1. Длину очереди типа int.
2. Указатель на первый элемент очереди типа Queue.

Сама функция имеет тип void так, как входе её работы функция не возвращает значений.

void AddElement(int size, Queue \*&first)

3. Для решения задачи данные были представлены в виде множества элементов типа char.

4. Реализация операций ввода и вывода данных:

* Ввод данных с консоли реализован при помощи цикла for и оператора cin.

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

Queue\* q = new Queue;

cout << "Введите " << i << " элемент: ";

cin >> el;

q->data = el;

p->next = q;

p = p->next;

}

p->next = nullptr;

* Вывод данных на консоль реализован при помощи цикла while и оператора cout.

while (p != nullptr) {

cout << p->data << " ";

p = p->next;

}

5. Поставленные задачи будут решены при помощи следующих действий:

* Программа запрашивает у пользователя длину двунаправленного списка.

while (size < 0) {

cin >> size;

if (size < 0) {

cout << "Введите размер очереди: ";

}

}

* При помощи функции FillQueue происходит заполнение очереди с консоли.

void FillQueue(int N, Queue \*&first) {

Queue\* p = new Queue;

char el;

cout << "Введите 1 элемент: ";

cin >> el;

p->data = el;

first = p;

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

Queue\* q = new Queue;

cout << "Введите " << i << " элемент: ";

cin >> el;

q->data = el;

p->next = q;

p = p->next;

}

p->next = nullptr;

};

* При помощи функции AddElement происходит добавления нового элемента перед элементом с указанным индексом. Программа запрашивает у пользователя номер элемента, перед которым ставится новый элемент, а затем с клавиатуры вводится новый элемент. При помощи оператора for указатель доходит до введённого номера элемента. Новый элемент подставляется путем изменения адресов элементов.

void AddElement(int size, Queue \*&first) {

int N = -1,count=0;

char key;

Queue\* p = first;

while ((N > size) || (N < 1)) {

if ((N > size) || (N < 1)) {

cout << "Введите номер элемента: ";

}

cin >> N;

}

cout << "Введите элемента который хотите добавить: ";

cin >> key;

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

p = p->next;

}

if (N != 1) {

Queue\* q = new Queue;

q->data = key;

q->next = p->next;

p->next = q;

q = q->next;

}

else {

Queue\* q = new Queue;

q->data = key;

q->next = first;

first = q;

}

};

* При помощи функции PrintQueue программа выводит элементы списка на консоль.

void PrintQueue(Queue\* first) {

Queue\* p = first;

while (p != nullptr) {

cout << p->data << " ";

p = p->next;

}

cout << endl <<endl;

};

Очередь через STL

1. Определить какие предстоит выполнить действия:

* Создать очередь через STL.

queue <char> Queue;

* Разработать функцию FillQueue для заполнения очереди.

void FillQueue(int N) {

char el;

for (int i = 1; i <= N; i++) {

cout << "Введите " << i << " элемент: ";

cin >> el;

Queue.push(el);

}

}

* Создать массив в котором буду хранится элементы очереди с добавлением нового элемента.

char\* array = new char[size+1];

* Разработать функцию AddElement для добавления в очередь элемента с заданным номером.

void AddElement(char arr[] ,int size) {

char key;

int N = -1;

while ((N < 1) || (N > size)) {

if ((N < 1) || (N > size)) {

cout << "\nВведите номер элемента: ";

}

cin >> N;

}

for (int i = 1; i <= size; i++) {

arr[i] = Queue.front();

Queue.pop();

}

for (int i = 1; i < N; i++) {

Queue.push(arr[i]);

}

cout << "\nВведите элемент: ";

cin >> key;

Queue.push(key);

for (int i = N; i <= size; i++) {

Queue.push(arr[i]);

}

}

2. С какими типами данных действие надо сделать, в каком виде эти данные будут представлены:

* Для запоминания длины очереди используем переменную size типа int.

int size=-1;

* Для инициализации очереди используем переменную Queue.

queue <char> Queue;

* Для обращения к динамическому массиву используем указатель array типа char.

char\* array = new char[size+1];

* Для хранения нового элемента используем переменную key типа char.

char key;

* Для функции FillList используем следующие элементы.

1. Длину очереди типа int.

Сама функция имеет тип void так, как входе её работы функция не возвращает значений.

void FillList(int N)

* Для функции AddElement используем следующие элементы.

1. Массив типа char
2. Длину очереди типа int.

Сама функция имеет тип void так, как входе её работы функция не возвращает значений.

void AddElement(char arr[] ,int size)

* Для функции PrintQueue используем следующие элементы.

1. Очередь

Сама функция имеет тип void так, как входе её работы функция не возвращает значений.

void PrintQueue(queue<char> Queue)

3. Для решения задачи данные были представлены в виде множества элементов типа char.

4. Реализация операций ввода и вывода данных:

* Ввод данных с консоли реализован при помощи цикла for и оператора cin.

for (int i = 1; i <= N; i++) {

cout << "Введите " << i << " элемент: ";

cin >> el;

Queue.push(el);

}

* Вывод данных на консоль реализован при помощи цикла for и оператора cout.

for (int i = 1; i <= N; i++) {

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

Queue.pop();

}

5. Поставленные задачи будут решены при помощи следующих действий:

* Программа запрашивает у пользователя размер очереди.

while (size < 0) {

if (size < 0) {

cout << "Введите размер очереди: ";

cin >> size;

}

}

* Создается динамический массив в котором будут хранится элементы очереди с добавлением нового элемента.

char\* array = new char[size+1];

* При помощи функции FillQueue происходит заполнение очереди с консоли.

void FillQueue(int N) {

char el;

for (int i = 1; i <= N; i++) {

cout << "Введите " << i << " элемент: ";

cin >> el;

Queue.push(el);

}

}

* При помощи функции AddElement происходит добавления нового элемента перед элементом с указанным индексом. Программа запрашивает у пользователя номер элемента, перед которым ставится новый элемент, а затем с клавиатуры вводится новый элемент. При помощи цикла for передаем элементы из очереди в массив при помощи функции front и удаляем элементы из очереди при помощи функции pop. Использую цикл for заполняем очередь элементами массива до указанного номера используя функцию push. Новый элемент подставляется при помощи функции push в очередь. Очередь заполняется оставшимися элементами массива через цикл for.

void AddElement(char arr[] ,int size) {

char key;

int N = -1;

while ((N < 1) || (N > size)) {

if ((N < 1) || (N > size)) {

cout << "\nВведите номер элемента: ";

}

cin >> N;

}

for (int i = 1; i <= size; i++) {

arr[i] = Queue.front();

Queue.pop();

}

for (int i = 1; i < N; i++) {

Queue.push(arr[i]);

}

cout << "\nВведите элемент: ";

cin >> key;

Queue.push(key);

for (int i = N; i <= size; i++) {

Queue.push(arr[i]);

}

}

* При помощи функции PrintQueue программа выводит элементы очереди на консоль.

void PrintQueue(queue<char> Queue) {

int N = Queue.size();

for (int i = 1; i <= N; i++) {

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

Queue.pop();

}

cout << endl;

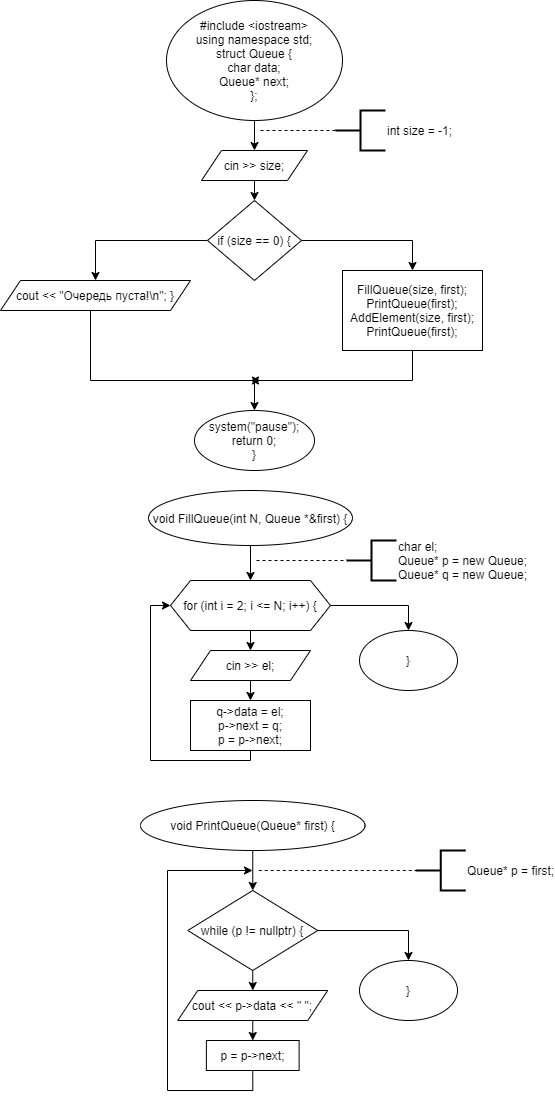
}

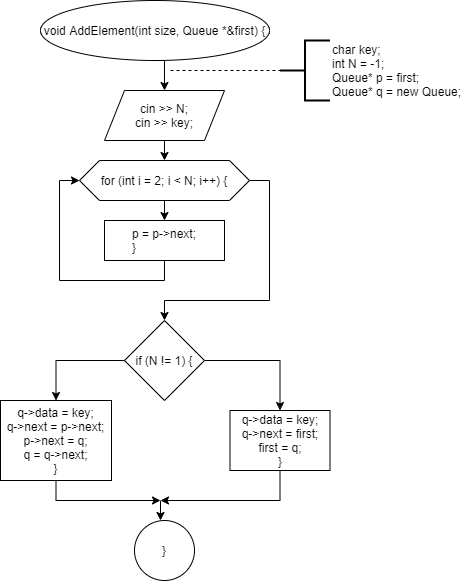
* Происходит удаление ранее созданного динамического массива array.

delete[]array;

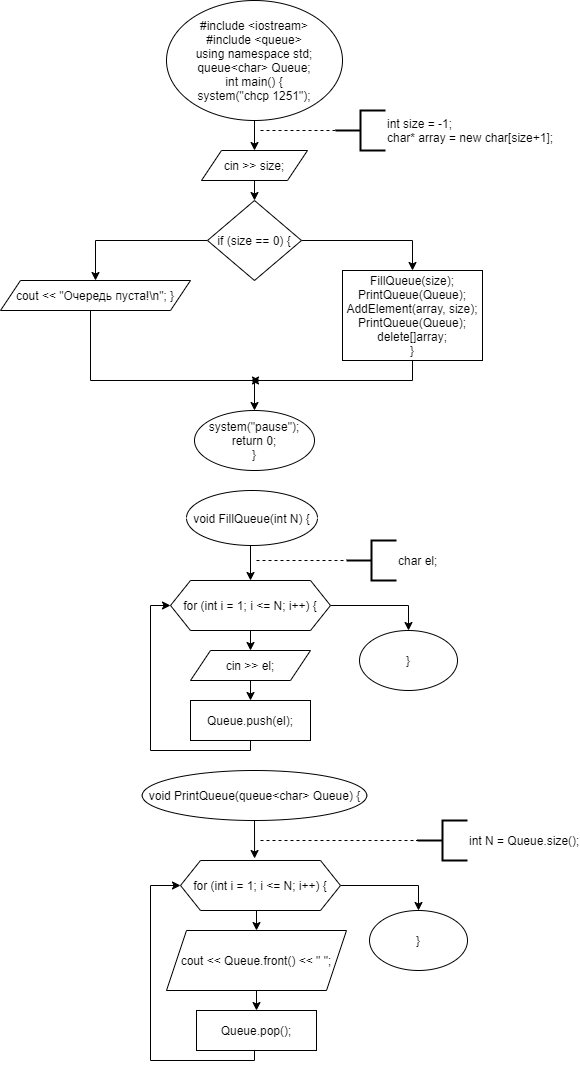
# **Блок схема**

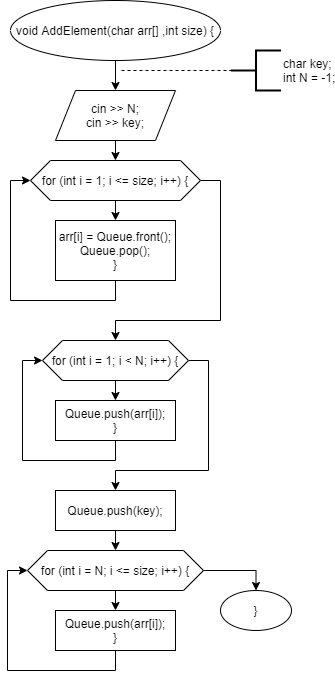
Очередь





Очередь через STL





# **Код программы на языке C++**

Очередь

#include <iostream>

using namespace std;

struct Queue {

char data;

Queue\* next;

};

Queue\* FillQueue(int N, Queue \*&first) {

Queue\* p = new Queue;

char el;

cout << "Введите 1 элемент: ";

cin >> el;

p->data = el;

first = p;

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

Queue\* q = new Queue;

cout << "Введите " << i << " элемент: ";

cin >> el;

q->data = el;

p->next = q;

p = p->next;

}

p->next = nullptr;

return first;

};

void PrintQueue(Queue\* first) {

Queue\* p = first;

while (p != nullptr) {

cout << p->data << " ";

p = p->next;

}

cout << endl <<endl;

};

void AddElement(int size, Queue \*&first) {

int N = -1;

char key;

Queue\* p = first;

while ((N > size) || (N < 1)) {

if ((N > size) || (N < 1)) {

cout << "Введите номер элемента: ";

}

cin >> N;

}

cout << "Введите элемента который хотите добавить: ";

cin >> key;

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

p = p->next;

}

if (N != 1) {

Queue\* q = new Queue;

q->data = key;

q->next = p->next;

p->next = q;

q = q->next;

}

else {

Queue\* q = new Queue;

q->data = key;

q->next = first;

first = q;

}

};

int main()

{

system("chcp 1251");

int size = -1;

Queue\* first;

while (size < 0) {

if (size < 0) {

cout << "Введите размер очереди: ";

}

cin >> size;

}

cout << endl;

if (size == 0) {

cout << "Очередь пуста!\n";

}

else {

FillQueue(size, first);

cout << "\nОчередь:\n";

PrintQueue(first);

AddElement(size, first);

cout << "\nОчередь:\n";

PrintQueue(first);

}

system("pause");

return 0;

}

Очередь через STL

#include <iostream>

#include <queue>

using namespace std;

queue<char> Queue;

void FillQueue(int N) {

char el;

for (int i = 1; i <= N; i++) {

cout << "Введите " << i << " элемент: ";

cin >> el;

Queue.push(el);

}

}

void PrintQueue(queue<char> Queue) {

int N = Queue.size();

for (int i = 1; i <= N; i++) {

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

Queue.pop();

}

cout << endl;

}

void AddElement(char arr[] ,int size) {

char key;

int N = -1;

while ((N < 1) || (N > size)) {

if ((N < 1) || (N > size)) {

cout << "\nВведите номер элемента: ";

}

cin >> N;

}

for (int i = 1; i <= size; i++) {

arr[i] = Queue.front();

Queue.pop();

}

for (int i = 1; i < N; i++) {

Queue.push(arr[i]);

}

cout << "\nВведите элемент: ";

cin >> key;

Queue.push(key);

for (int i = N; i <= size; i++) {

Queue.push(arr[i]);

}

}

int main()

{

system("chcp 1251");

int size = -1;

while (size < 0) {

if (size < 0) {

cout << "Введите размер очереди: ";

cin >> size;

}

}

cout << endl;

if (size == 0) {

cout << "Очередь пуста!\n";

}

else {

char\* array = new char[size];

FillQueue(size);

cout << "\nОчередь:\n";

PrintQueue(Queue);

AddElement(array, size);

cout << "\nОчередь после добавления:\n";

PrintQueue(Queue);

delete [] array;

}

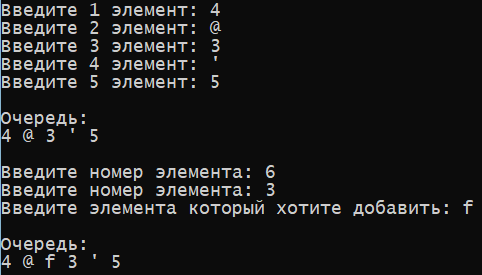
system("pause");

return 0;

}

# **Скриншоты тестов**

Очередь



Очередь через STL

