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 №11.2**

Дисциплина: Основы алгоритмизации и программирования

Тема: “Динамические структуры данных”

Вариант 15

Выполнил:

Студент гр. ИВТ-20-2б

Чувашев Максим Алексеевич

Проверил:

Доцент кафедры ИТАС  
Полякова О.А

Пермь, 2021

**Цель работы**

Получить практические навыки работы с очередями;

**Постановка задачи**

1. Сформировать очередь. Тип информационного поля указан в варианте. (Тип информационного поля int. Удалить из списка все элементы с четными номерами.)
2. Распечатать полученную структуру.
3. Выполнить обработку структуры в соответствии с заданием.
4. Распечатать полученный результат.

**Анализ задачи**

1. Используемые типы данных.
   1. Структура Queue используется для хранения информационного поля типа int и для хранения адреса на следующий элемент очереди.

struct Queue

{

int inf; // информационное поле

Queue\* next; // поле, которое хранит адрес следующего элемента очереди

};

* 1. Тип int используется как счетчик, как параметр в цикле for, а также как информационное поле в очереди.

int inf;

int count = 1; // создание счетчика

for (int i = 2; i <= size; i++)

* 1. Тип queue<int> используется для хранения очереди при реализации очереди через библиотеки STL.

1. Действия над используемыми данными.
   1. С данными типа int:
      1. Инкремент

count++; // увеличиваем счетчик

* + 1. Сравнение

if (count % 2 == 0)

* 1. С данными типа структуры Queue:
     1. Создание очереди
     2. Удаление четных элементов очереди
     3. Вывод всех элементов в консоль
  2. С данными типа queue<int>:
     1. Создание очереди
     2. Удаление четных элементов очереди
     3. Вывод всех элементов в консоль

1. Вид данных.
   1. Очередь реализована в программе с помощью структуры, которая имеет одно информационное поле типа int и еще одно поле, которое хранит указатель на следующий элемент. Так же очередь реализована с помощью библиотеки STL.
2. Структура.
   1. В программе используется структура Queue, которая имеет два поля, одно информационное типа int и второе поле, которое хранит адрес следующего элемента очереди.

struct Queue

{

int inf; // информационное поле

Queue\* next; // поле, которое хранит адрес следующего элемента очереди

};

1. Ввод и вывод.
   1. Ввод и вывод всех данных осуществляется в консоль с помощью операторов ввода и вывода cin и cout.

cout << "Введите размер очереди: \t";

cin >> (\*size);

1. Действия для решения задачи.
   1. Реализация структуры, которая имеет два поля, одно информационное типа int и второе поле, которое хранит адрес следующего элемента очереди.

struct Queue

{

int inf; // информационное поле

Queue\* next; // поле, которое хранит адрес следующего элемента очереди

};

* 1. Функция ввода размера очереди реализована как через структуры, так и через библиотеки STL идентично.

void EnterSize(int\* size) // функция ввода размера очереди

{

cout << "Введите размер очереди: \t";

cin >> (\*size);

while ((\*size) < 1)

{

cout << "Введен неправильный размер очереди!\n";

cout << "Введите размер очереди: \t";

cin >> (\*size);

}

}

* 1. Создание очереди.
     1. Реализация через структуры. Сначала очередь проверяется на существование. Далее в цикле создаются элементы очереди и помещаются в очередь. Попутно они связываются между собой с помощью поля, которое будет указывать на следующий элемент. Подробнее описано в комментариях к коду.

Queue\* InitQueue(Queue\* &ptrFirst, int size) // создание очереди

{

if (size == 0) // проверка на существование очереди

{

return NULL;

}

else

{

Queue\* ptr = new Queue; // создание элемента очереди

ptrFirst = ptr; // указатель на первый элемент очереди срановится равным новому элементу

ptr->next = NULL; // поле хранящее адрес следующего элемента, равен null

cout << "Введите 1 элемент очереди:\t";

cin >> ptr->inf; // пользователь вводит данные информационного поля

for (int i = 2; i <= size; i++) // цикл от 2 до размера списка

{

Queue\* h = new Queue; // создание нового элемента очереди

cout << "Введите " << i << " элемент очереди:\t";

cin >> h->inf; // пользователь вводит данные информационного поля

h->next = NULL; // указатель нового элемента на след элемент равен null

ptr->next = h; // предыдущий элемент связывается со следующим

ptr = ptr->next; // переходим к следующему элементу

}

}

return ptrFirst; // возвращаем указатель на первый элемент очереди

}

* + 1. Реализация через библиотеки STL. Создается временная переменная, которая будет хранить элементы очереди поочередно. Проходимся циклом по всему размеру очереди. Пользователь вводит элемент очереди. Помещаем его в конец очереди. Подробнее в комментариях к коду.

void InitQueue(int size, queue<int> &q) // функция создания очереди

{

int a; // создание временной переменной которая будет хранить значение одного из элементов очереди

for (int i = 1; i <= size; i++) // цикл от 0 до размера списка

{

cout << "Введите " << i << " элемент очереди:\t";

cin >> a; // ввод элемента списка

q.push(a); // помещение элемента списка в конец очереди

}

}

* 1. Печаль всех элементов очереди в консоль.
     1. Реализация через структуры. Создается временный указатель, который приравнивается первому элементу очереди. Далее пока данный указатель не будет равен NULL будут выводится все элементы очереди, а указатель будет передвигаться на следующий элемент очереди. Подробнее в комментариях к коду.

void PrintQueue(Queue\* ptrFirst, int size) // функция вывода в консоль списка

{

cout << endl;

Queue\* ptr = ptrFirst; // создание указателя на первый элемент

while(ptr != NULL) // цикл пока есть элемент

{

cout << ptr->inf << " "; // вывод элемета в консоль

ptr = ptr->next; // переход на следующий элемент очереди

}

cout << endl;

}

* + 1. Реализация через библиотеки STL. Выводятся все элементы очереди пока очередь не окажется пуста. Подробнее в комментариях к коду.

void PrintQueue(queue<int> q) // печать всех элементов очереди в консоль

{

cout << endl;

while (!q.empty()) // пока список не пуст

{

cout << q.front() << " "; // вывод элемента списка в консоль

q.pop(); // удаление первого элемента списка из очереди

}

cout << endl;

}

* 1. Функция удаления каждого четного элемента очереди.
     1. Реализация через структуры. Создается счетчик. Проходимся циклом по всем элементам очереди. Если элемент четный. Запоминаем его в отдельную переменную. Перестраиваем связи, исключая его из цепочки. Следом благополучно удаляем его. Подробнее в комментариях к коду.

void EditQueue(Queue\* ptrFirst, int size) // удаление четных элементов очереди

{

int count = 1; // создание счетчика

Queue\* ptr = ptrFirst; // создание временной переменной равный перввому элементу списка

Queue\* old = ptr; // временная переменная, которая будет хранить старый элемент списка

Queue\* D; // создание временного указателя

while (ptr != NULL) // пока есть элементы списка

{

if (count % 2 == 0) // если элемент четный

{

D = ptr; // приравниваем указатель D к временной переменной

ptr = ptr->next; // переходим к следующему элементу

old->next = ptr; // указываем в поле со следующим элементом на новый элемент

delete D; // удаляем указатель D

}

else

{

old = ptr; // запоминаем новый элемент

ptr = ptr->next; // переходим на следующий элемент

}

count++; // увеличиваем счетчик

}

}

* + 1. Реализация через библиотеки STL. Создается счетчик. Создается временная очередь. Пока изначальная очередь не пуста проходимся по ней. Если элемент четный, то добавляем его в новую очередь. Достаем элемент из изначальной очереди. И в конце приравниваем получившуюся очередь изначальной. Подробнее в комментариях к коду.

void EditQueue(queue<int> &q) // функция удаления каждого второго элемента очереди

{

int count = 1; // создаение счетчика

queue<int> qd; // создание временной очереди

while (!q.empty()) // проверка на пустоту очереди

{

if (count % 2 != 0) // если элемент нечетный

{

qd.push(q.front()); // то помещаем его в новую очередь

}

q.pop(); // удаляем первый элемент изначальной очереди

count++; // увеличиваем счетчик

}

q = qd; // приравниваем временную очередь к старой очереди

}

* 1. Функция main()
     1. Реализация через структуры. Последовательно вызываются все функции.

int main()

{

setlocale(LC\_ALL, "Ru");

int size;

EnterSize(&size);

Queue\* ptrFirst;

ptrFirst = InitQueue(ptrFirst, size);

PrintQueue(ptrFirst, size);

EditQueue(ptrFirst, size);

PrintQueue(ptrFirst, size);

}

* + 1. Реализация через библиотеки STL. Последовательно вызываются все функции.

int main()

{

setlocale(LC\_ALL, "Ru");

int size;

EnterSize(&size);

queue<int> q;

InitQueue(size, q);

PrintQueue(q);

EditQueue(q);

PrintQueue(q);

}

**Полный код реализация через структуры**

#include <iostream>

using namespace std;

struct Queue

{

int inf; // информационное поле

Queue\* next; // поле, которое хранит адрес следующего элемента очереди

};

void EnterSize(int\* size) // функция ввода размера очереди

{

cout << "Введите размер очереди: \t";

cin >> (\*size);

while ((\*size) < 1)

{

cout << "Введен неправильный размер очереди!\n";

cout << "Введите размер очереди: \t";

cin >> (\*size);

}

}

Queue\* InitQueue(Queue\* &ptrFirst, int size) // создание очереди

{

if (size == 0) // проверка на существование очереди

{

return NULL;

}

else

{

Queue\* ptr = new Queue; // создание элемента очереди

ptrFirst = ptr; // указатель на первый элемент очереди срановится равным новому элементу

ptr->next = NULL; // поле хранящее адрес следующего элемента, равен null

cout << "Введите 1 элемент очереди:\t";

cin >> ptr->inf; // пользователь вводит данные информационного поля

for (int i = 2; i <= size; i++) // цикл от 2 до размера списка

{

Queue\* h = new Queue; // создание нового элемента очереди

cout << "Введите " << i << " элемент очереди:\t";

cin >> h->inf; // пользователь вводит данные информационного поля

h->next = NULL; // указатель нового элемента на след элемент равен null

ptr->next = h; // предыдущий элемент связывается со следующим

ptr = ptr->next; // переходим к следующему элементу

}

}

return ptrFirst; // возвращаем указатель на первый элемент очереди

}

void PrintQueue(Queue\* ptrFirst, int size) // функция вывода в консоль списка

{

cout << endl;

Queue\* ptr = ptrFirst; // создание указателя на первый элемент

while(ptr != NULL) // цикл пока есть элемент

{

cout << ptr->inf << " "; // вывод элемета в консоль

ptr = ptr->next; // переход на следующий элемент очереди

}

cout << endl;

}

void EditQueue(Queue\* ptrFirst, int size) // удаление четных элементов очереди

{

int count = 1; // создание счетчика

Queue\* ptr = ptrFirst; // создание временной переменной равный перввому элементу списка

Queue\* old = ptr; // временная переменная, которая будет хранить старый элемент списка

Queue\* D; // создание временного указателя

while (ptr != NULL) // пока есть элементы списка

{

if (count % 2 == 0) // если элемент четный

{

D = ptr; // приравниваем указатель D к временной переменной

ptr = ptr->next; // переходим к следующему элементу

old->next = ptr; // указываем в поле со следующим элементом на новый элемент

delete D; // удаляем указатель D

}

else

{

old = ptr; // запоминаем новый элемент

ptr = ptr->next; // переходим на следующий элемент

}

count++; // увеличиваем счетчик

}

}

int main()

{

setlocale(LC\_ALL, "Ru");

int size;

EnterSize(&size);

Queue\* ptrFirst;

ptrFirst = InitQueue(ptrFirst, size);

PrintQueue(ptrFirst, size);

EditQueue(ptrFirst, size);

PrintQueue(ptrFirst, size);

}

**Полный код реализация через библиотеки STL**

#include <iostream>

#include <queue>

using namespace std;

void EnterSize(int\* size) // фукнция ввода размера очереди

{

cout << "Введите размер очереди: \t";

cin >> \*size;

while (\*size < 1)

{

cout << "Введен некорректный размер очереди!" << endl;

cout << "Введите размер очереди: \t";

cin >> \*size;

}

}

void InitQueue(int size, queue<int> &q) // функция создания очереди

{

int a; // создание временной переменной которая будет хранить значение одного из элементов очереди

for (int i = 1; i <= size; i++) // цикл от 0 до размера списка

{

cout << "Введите " << i << " элемент очереди:\t";

cin >> a; // ввод элемента списка

q.push(a); // помещение элемента списка в конец очереди

}

}

void PrintQueue(queue<int> q) // печать всех элементов очереди в консоль

{

cout << endl;

while (!q.empty()) // пока список не пуст

{

cout << q.front() << " "; // вывод элемента списка в консоль

q.pop(); // удаление первого элемента списка из очереди

}

cout << endl;

}

void EditQueue(queue<int> &q) // функция удаления каждого второго элемента очереди

{

int count = 1; // создаение счетчика

queue<int> qd; // создание временной очереди

while (!q.empty()) // проверка на пустоту очереди

{

if (count % 2 != 0) // если элемент нечетный

{

qd.push(q.front()); // то помещаем его в новую очередь

}

q.pop(); // удаляем первый элемент изначальной очереди

count++; // увеличиваем счетчик

}

q = qd; // приравниваем временную очередь к старой очереди

}

int main()

{

setlocale(LC\_ALL, "Ru");

int size;

EnterSize(&size);

queue<int> q;

InitQueue(size, q);

PrintQueue(q);

EditQueue(q);

PrintQueue(q);

}

**Блок схема реализация через структуры**













**Блок схема реализация через STL**



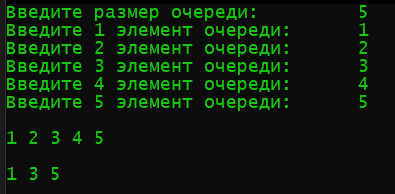
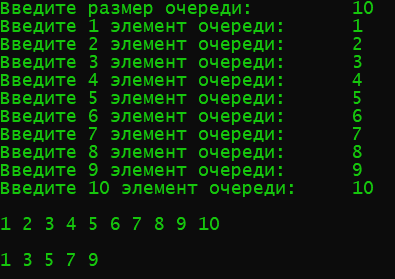
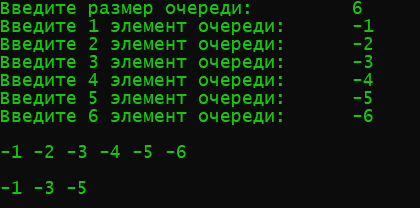
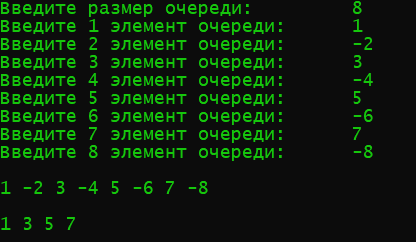
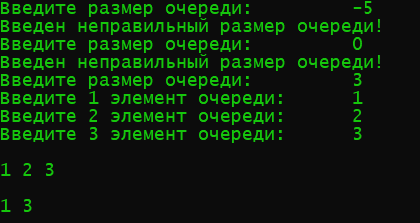








**Скриншоты результатов работы программы**

1. Нормальный ввод:
   1. Реализация через структуры.
      1. 
      2. 
   2. Реализация через библиотеку STL.
      1. 
      2. 
2. Некорректный ввод
   1. Реализация через структуры.
      1. 
   2. Реализация через библиотеки STL
      1. 