Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе №11.2**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: Информационные и динамические структуры. Очереди

Вариант 19

Выполнил:

Студент группы РИС-20-1б

Ремянников Александр Владимирович

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

**Пермь, 2021**

**Цель работы**

Получить практические навыки работы с очередями.

**Постановка задачи**

(Вариант 19)

1. Сформировать однонаправленный и двунаправленный списки или стек и очередь.

2. Распечатать полученную структуру.

3. Выполнить обработку структуры в соответствии с заданием.

3.1. (Очередь) Добавить в список элементы с номерами 1, 3, 5 и т. д.

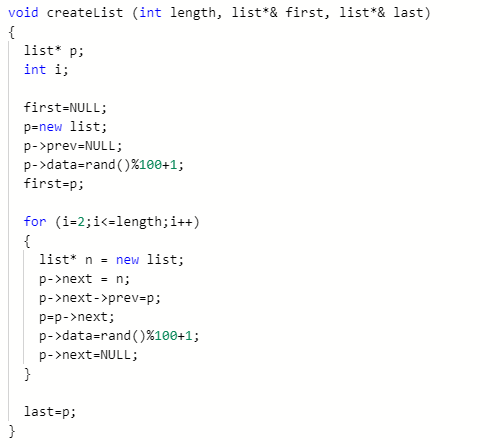
4. Распечатать полученный результат.

5. Удалить соответствующую структуру из памяти.

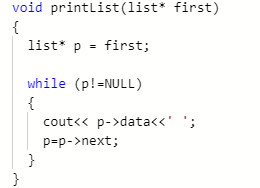
**Анализ задачи (Очередь, указатели)**

**1.** Для решения задачи необходимо:

**1.1**. Реализовать функцию для создания очереди



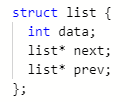
**1.2.** Для работы с пользователем предусмотреть вывод очереди на экран



**1.3.** Для добавления на нечетные позиции новых элементов реализовать функцию addOnOddPosition

**2.** В ходе работы были использованы следующие типы данных:

**2.1.** Для создания очереди используется структура list, тип int в которой предназначен для хранения значения элемента очереди, а тип list\* - для хранения информации о местонахождении в памяти следующего и предыдущего элемента очереди.

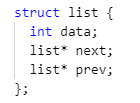


**2.2.** Для хранения адреса первого и последнего элемента очереди используется тип list\*



**3.** Для решения задачи данные были представлены в следующем виде:

**3.1.** Очередь была представлена в виде структуры с информационным и адресными полями.



**4.** Для операций ввода и вывода использовались следующие операторы и функции:

**4.1.** Для вывода использовался оператор cout.



**4.2.** Для создания очереди со случайными значениями использовалась функция rand().



**5.** Поставленные задачи будут решены следующими действиями:

**5.1.** Будет применена ранее созданная функция createList для создания очереди.



**5.2.** Для добавления на нечетные позиции новых элементов применена функция addOnOddPosition.



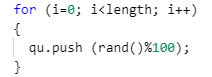
**Анализ задачи (Очередь, STL)**

**1.** Для решения задачи необходимо:

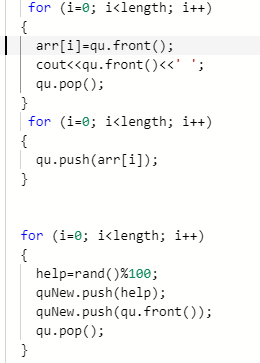
**1.1.** Для работы с двунаправленными списками необходимо подключить библиотеки queue и iterator



**1.2.** Заполнить очередь случайными значениями.



**1.3.** Реализовать код для добавления на нечетные позиции новых элементов.

****

**2.** В ходе работы были использованы следующие типы данных:

**2.1.** Для очереди был применен класс queue.



**3.** Для решения задачи данные были представлены в следующем виде:

**3.1.** Очередь была представлена в виде класса queue из библиотеки queue.



**4.** Для операций ввода и вывода использовались следующие операторы и функции:

**4.1.** Для вывода использовались оператор cout

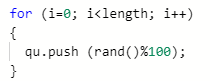


**4.2.** Для создания очереди со случайными значениями использовалась функция rand().

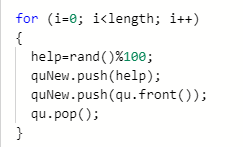


**5.** Поставленные задачи будут решены следующими действиями:

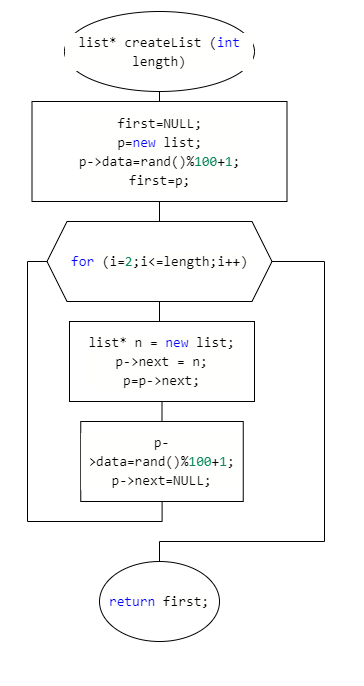
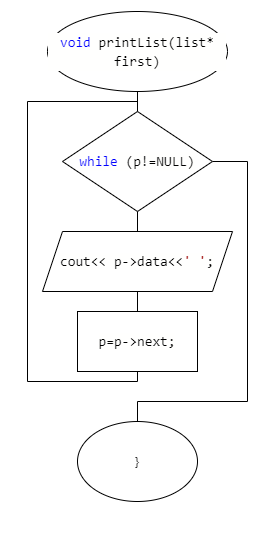
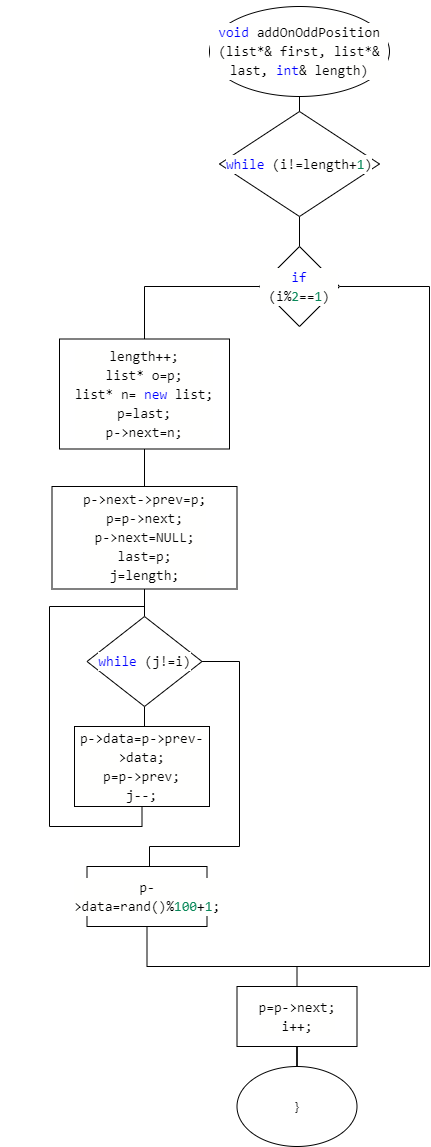
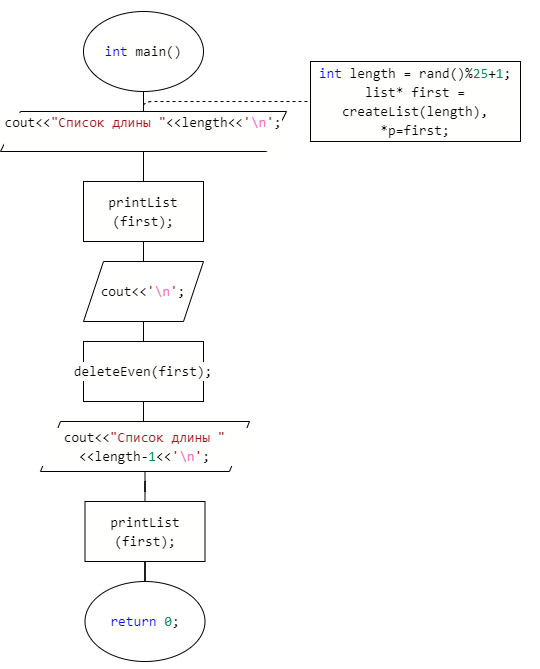
**5.1.** Для создания очереди будет применена функция для добавления элемента в очередь внутри цикла for.



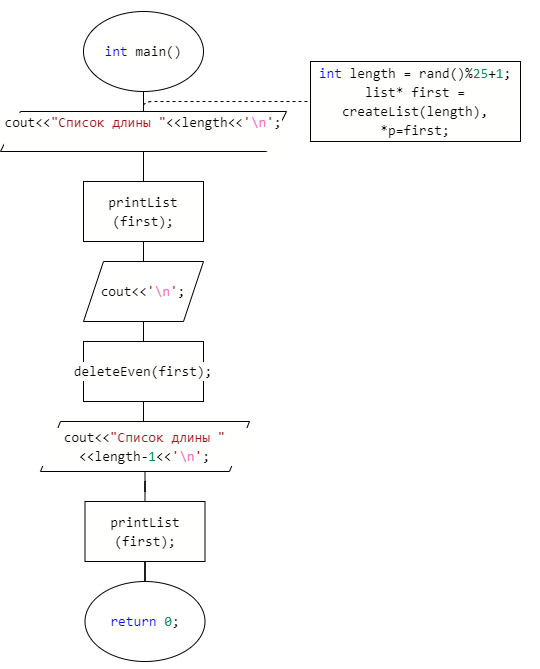
**5.2.** Для добавления элементов будет применен цикл for внутри которого будут добавлены элементы так, что очередь будет удовлетворять условию задачи.



**Блок-схема программы (Очередь, указатели)**

****

**Блок-схема программы (Очередь, STL)**

****

**Решение (Очередь, указатели)**

//Добавить в очередь элементы с номерами 1, 3, 5 и т. д.

#include <iostream>

using namespace std;

struct list {

int data;

list\* next;

list\* prev;

};

void createList (int, list\*&, list\*&);

void printList (list\*, int);

void deleteEven (list\*);

void addOnOddPosition (list\*&, list\*&, int&);

int main()

{

int length = rand()%25+1;

list\* first, \*last;

createList(length, first, last);

printList (first, length);

cout<<'\n';

addOnOddPosition (first, last, length);

printList (first, length);

return 0;

}

void createList (int length, list\*& first, list\*& last)

{

list\* p;

int i;

first=NULL;

p=new list;

p->prev=NULL;

p->data=rand()%100+1;

first=p;

for (i=2;i<=length;i++)

{

list\* n = new list;

p->next = n;

p->next->prev=p;

p=p->next;

p->data=rand()%100+1;

p->next=NULL;

}

last=p;

}

void printList(list\* first, int length)

{

list\* p = first;

cout<<"Очередь длины "<<length<<'\n';

while (p!=NULL)

{

cout<< p->data<<' ';

p=p->next;

}

}

void deleteEven (list\* first)

{

list\* p = first;

while (p->next!=NULL)

{

if (p->next->data % 2 == 0)

{

p->next=p->next->next;

break;

}

p=p->next;

}

}

void addOnOddPosition (list\*& first, list\*& last, int& length)

{

int i=1, j;

list\* p=first;

while (i!=length+1)

{

if (i%2==1)

{

length++;

list\* o=p;

list\* n= new list;

p=last;

p->next=n;

p->next->prev=p;

p=p->next;

p->next=NULL;

last=p;

j=length;

while (j!=i)

{

p->data=p->prev->data;

p=p->prev;

j--;

}

p->data=rand()%100+1;

}

p=p->next;

i++;

}

}

**Решение (Очередь, STL)**

#include <iostream>

#include <queue>

#include <iterator>

using namespace std;

int main ()

{

queue <int> qu;

queue <int> quNew;

int help, i, length = rand()%15+1, arr[50];

for (i=0; i<length; i++)

{

qu.push (rand()%100);

}

cout<<"Ваша очередь: ";

for (i=0; i<length; i++)

{

arr[i]=qu.front();

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

qu.pop();

}

for (i=0; i<length; i++)

{

qu.push(arr[i]);

}

for (i=0; i<length; i++)

{

help=rand()%100;

quNew.push(help);

quNew.push(qu.front());

qu.pop();

}

cout<<"\nВаша очередь: ";

for (i=0; i<(length\*2); i++)

{

arr[i]=quNew.front();

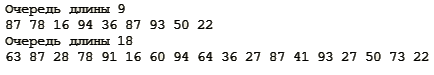
cout<<quNew.front()<<' ';

quNew.pop();

}

}

**Скриншоты результатов работы программы (Очередь, указатели)**

****

**Скриншоты результатов работы программы (Очередь, STL)**

****

**GitHub**

<https://github.com/sugarrrfqs/pnrpuLab11/tree/main>