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

Выполнил работу

студент группы РИС-20-1б

Ремянников А. В.

Проверила

Доцент кафедры ИТАС

к.т.н Полякова О. А.

Пермь, 2021

**Цель работы**

1) Получить практические навыки работы с однонаправленными списками;

2) Получить практические навыки работы с двунаправленными списками;

3) Получить практические навыки работы со стеками;

4) 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. (Однонаправленный список**)** Тип информационного поля int. Удалить из списка первый элемент с четным информационным полем.

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

3.3. (Стек) Тип информационного поля int. Удалить из списка первый элемент с четным информационным полем.

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

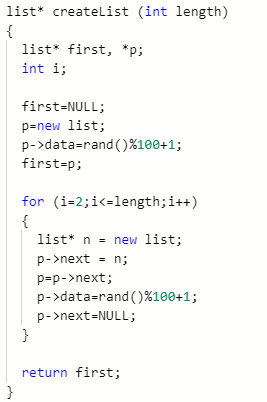
4. Распечатать полученный результат.

5. Удалить соответствующую структуру из памяти.

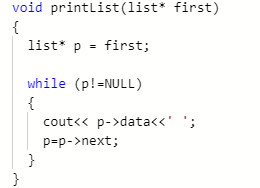
**Анализ задачи (Однонаправленный список)**

**1.** Для решения задачи необходимо:

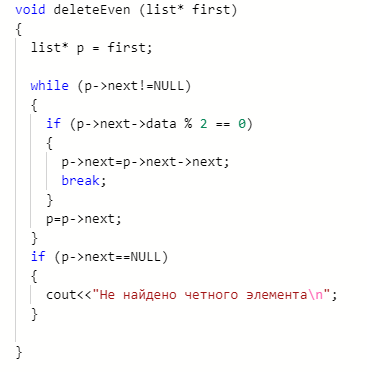
**1.1**. Реализовать функцию для создания списка



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

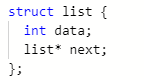


**1.3.** Реализовать функцию для удаления из списка первого элемента с четным информационным полем.



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

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

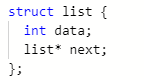


**2.2.** Для хранения адреса первого элемента списка используется тип list\*



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

**3.1.** Список был представлен в виде структуры с информационным и адресным полями.



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

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

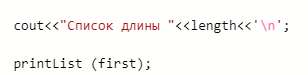


**4.2.** Для создания списка со случайными значениями использовалась функция rand().



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

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



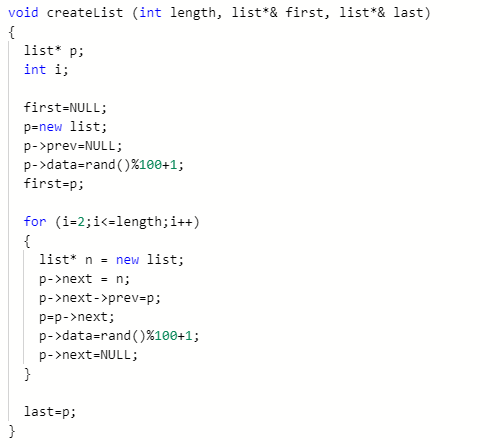
**5.2.** Для удаления из списка первого элемента с четным информационным полем используется функция deleteEven.



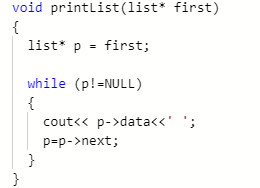
**Анализ задачи (Двунаправленный список, указатели)**

**1.** Для решения задачи необходимо:

**1.1**. Реализовать функцию для создания списка



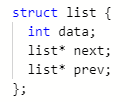
**1.2.** Для работы с пользователем предусмотреть вывод списка на экран



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

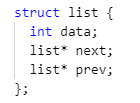


**2.2.** Для хранения адреса первого и последнего элемента списка используется тип list\*



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

**3.1.** Список был представлен 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.

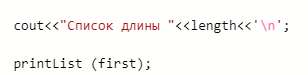


**4.2.** Для создания списка со случайными значениями использовалась функция rand().



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

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



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



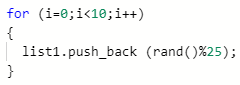
**Анализ задачи (Двунаправленный список, STL)**

**1.** Для решения задачи необходимо:

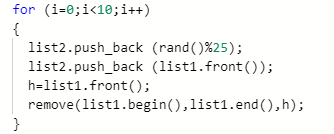
**1.1.** Для работы с двунаправленными списками необходимо подключить библиотеки <list> и <iterator>



**1.2.** Заполнить список 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 списков был применен класс list.



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

**3.1.** Список был представлен в виде класса list из библиотеки list.



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

**4.1.** Для вывода использовались операторы cout и copy



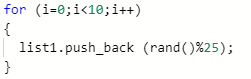


**4.2.** Для создания списка со случайными значениями использовалась функция rand().

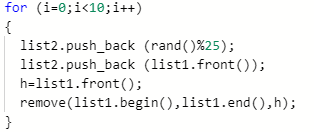


**5.** Поставленные задачи будут решены следующими действиями:

**5.1.** Для создания списка будет применена функция для добавления элемента в список внутри цикла for.



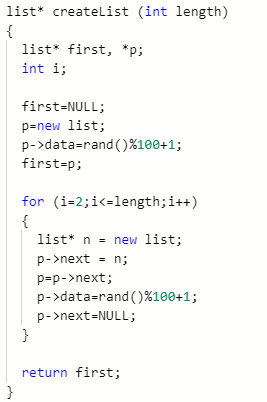
**5.2.** Для добавления элементов будет применен цикл for внутри которого будет создан новый список удовлетворяющий условию задачи.



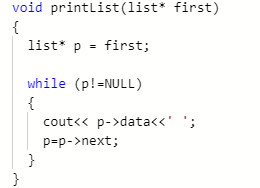
**Анализ задачи (Стек, Указатели)**

**1.** Для решения задачи необходимо:

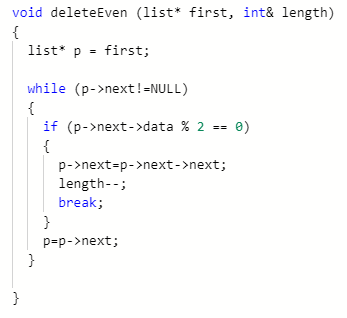
**1.1**. Реализовать функцию для создания стека



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

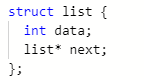


**1.3.** Реализовать функцию для удаления из стека первого элемента с четным информационным полем.



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

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

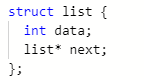


**2.2.** Для хранения адреса первого элемента стека используется тип list\*



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

**3.1.** Стек, как подтип однонаправленного списка был представлен в виде структуры с информационным и адресным полями.



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

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



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



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

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



**5.2.** Для удаления из списка первого элемента с четным информационным полем используется функция deleteEven.



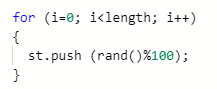
**Анализ задачи (Стек, STL)**

**1.** Для решения задачи необходимо:

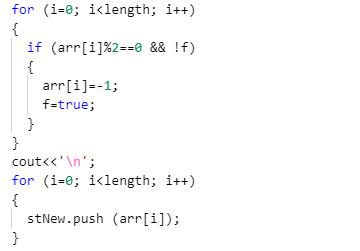
**1.1.** Для работы с двунаправленными списками необходимо подключить библиотеки stack и iterator.



**1.2.** Необходимо заполнить стек случайными значениями.



**1.3.** Реализовать код, позволяющий удалить из стека первый элемент с четным информационным полем.



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

**2.1.** Для работы со стеками используется класс stack



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

**3.1.** Стек был представлен в виде класса stack из библиотеки stack.



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

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

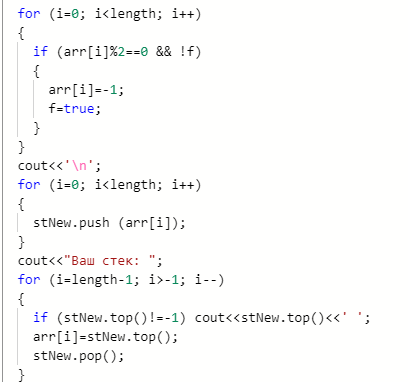


**4.2.** Для создания списка со случайными значениями использовалась функция rand().



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

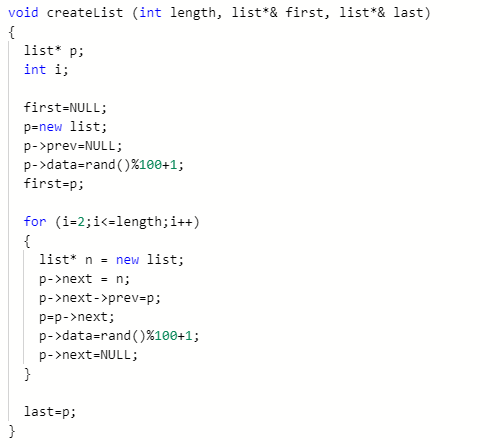
**5.1.** Будет применен код для удаления элемента с четным информационным полем, таким образом будет создан новый стек, который будет удовлетворять условиям задачи.



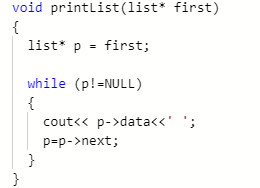
**Анализ задачи (Очередь, указатели)**

**1.** Для решения задачи необходимо:

**1.1**. Реализовать функцию для создания очереди



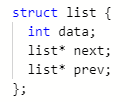
**1.2.** Для работы с пользователем предусмотреть вывод очереди на экран



**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.

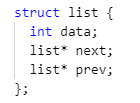


**2.2.** Для хранения адреса первого и последнего элемента очереди используется тип list\*



**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.



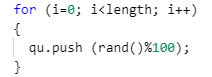
**Анализ задачи (Очередь, STL)**

**1.** Для решения задачи необходимо:

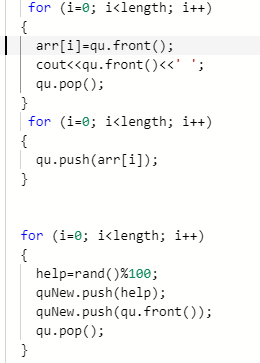
**1.1.** Для работы с двунаправленными списками необходимо подключить библиотеки queue и iterator



**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

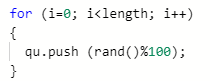


**4.2.** Для создания очереди со случайными значениями использовалась функция rand().

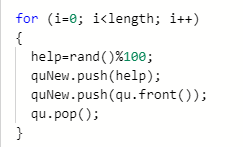


**5.** Поставленные задачи будут решены следующими действиями:

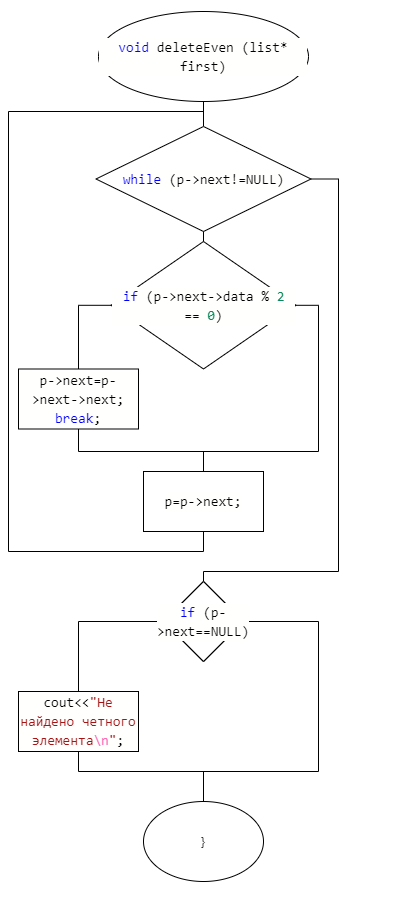
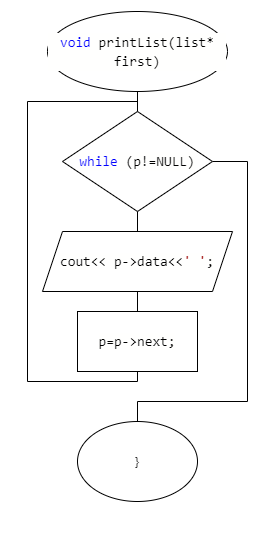
**5.1.** Для создания очереди будет применена функция для добавления элемента в очередь внутри цикла for.

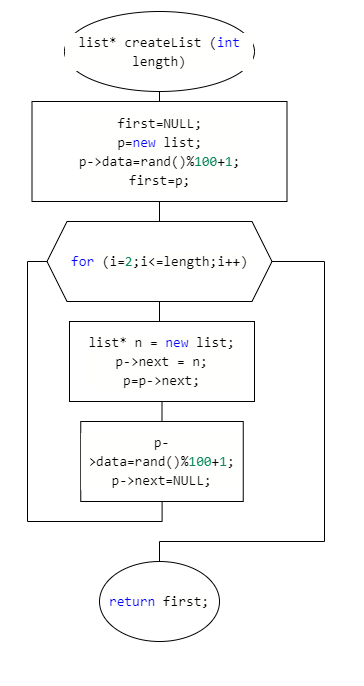
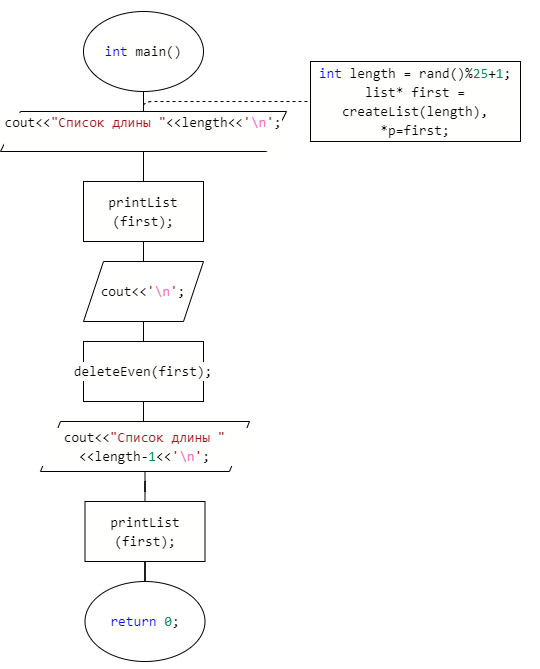


**5.2.** Для добавления элементов будет применен цикл for внутри которого будут добавлены элементы так, что очередь будет удовлетворять условию задачи.

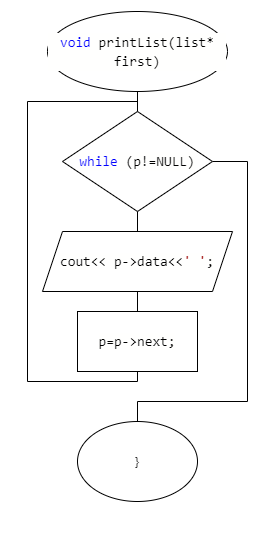
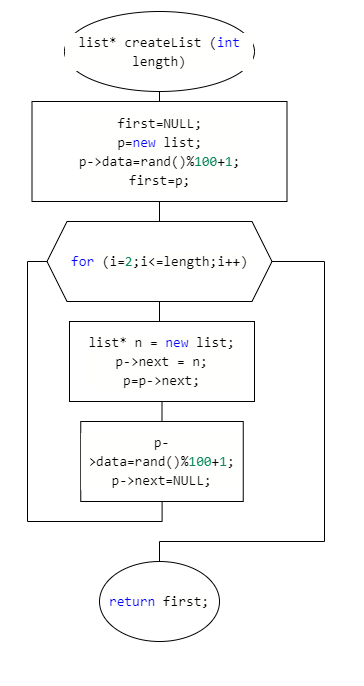
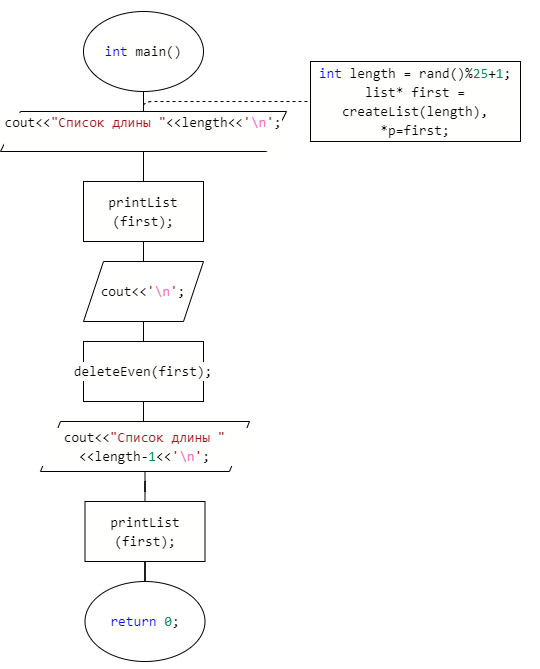


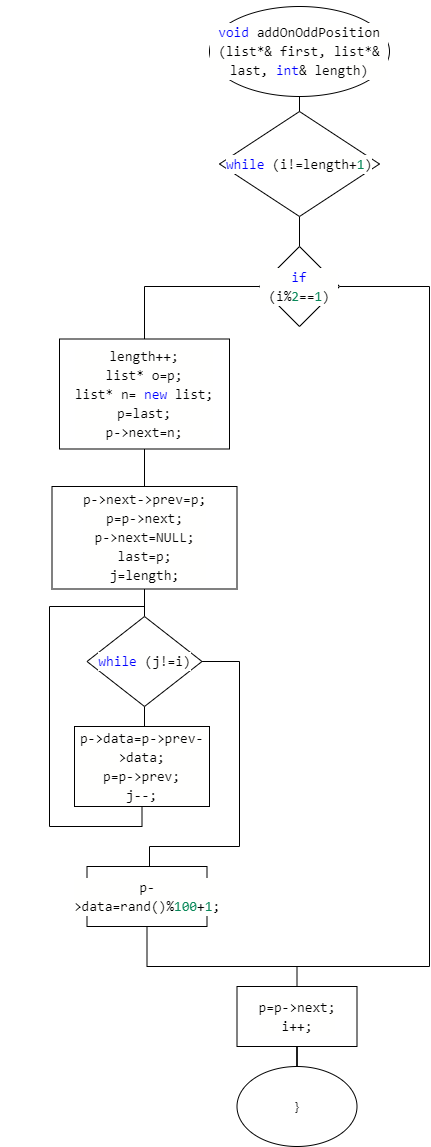
**Блок-схема программы (Однонаправленный список)**

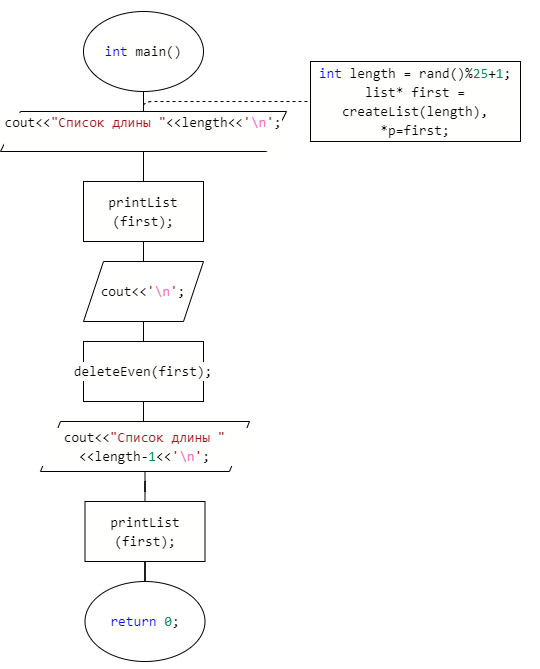
****

****

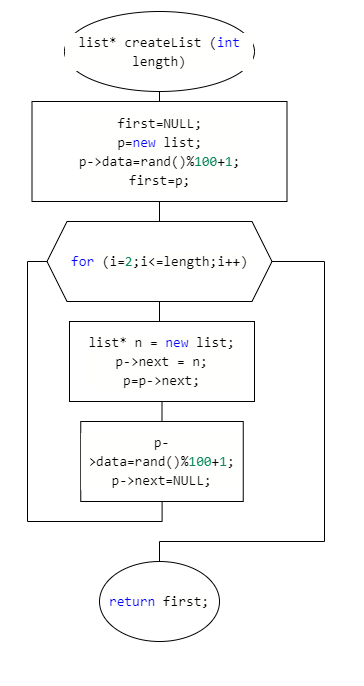
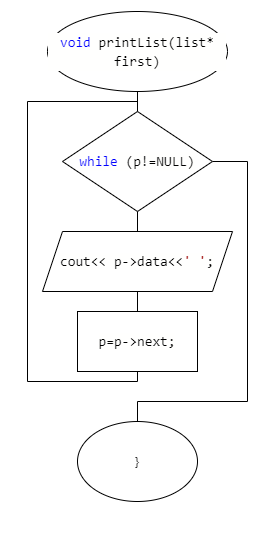
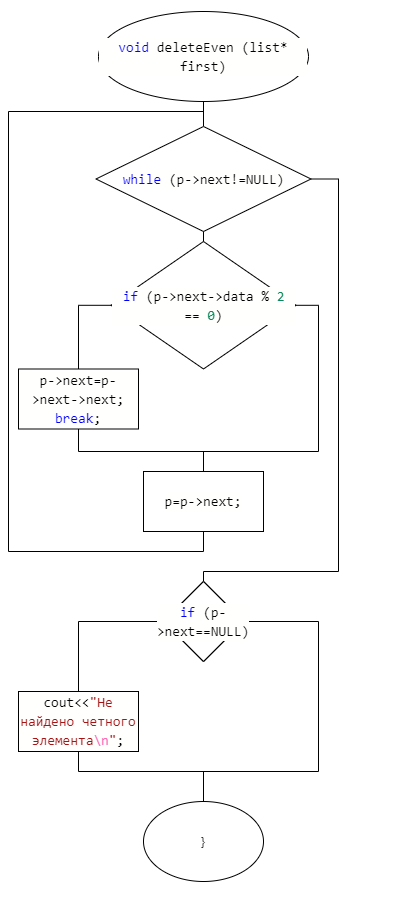
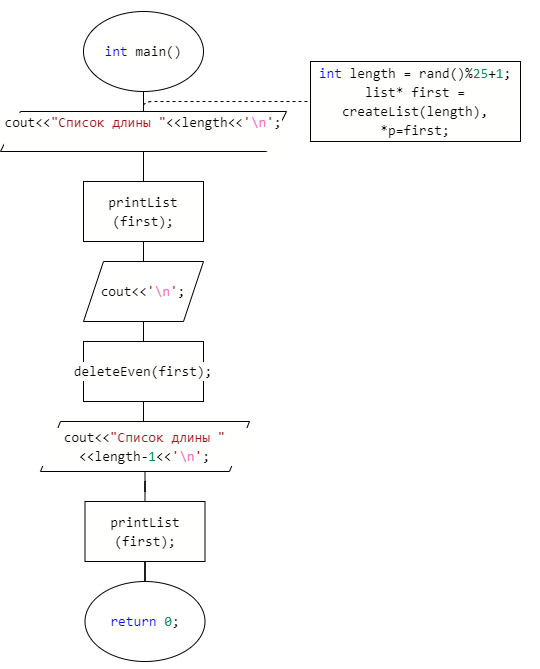
**Блок-схема программы (Двунаправленный список, указатели)**

****

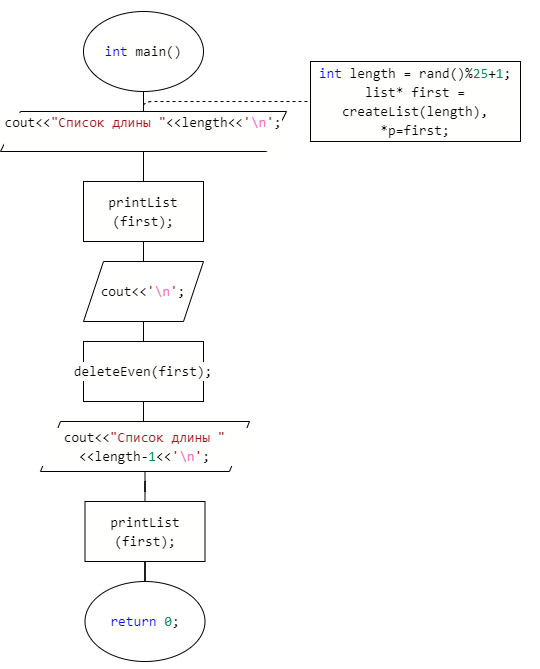
**Блок-схема программы (Двунаправленный список, STL)**

****

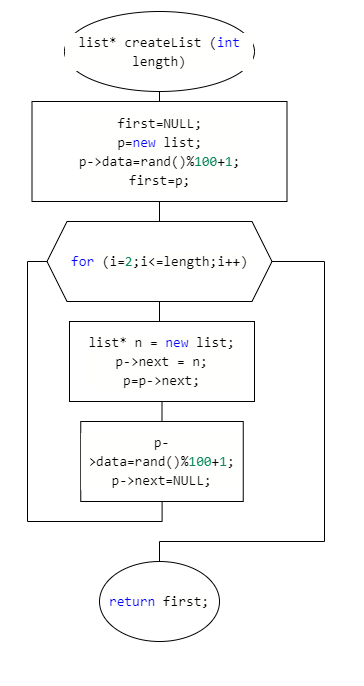
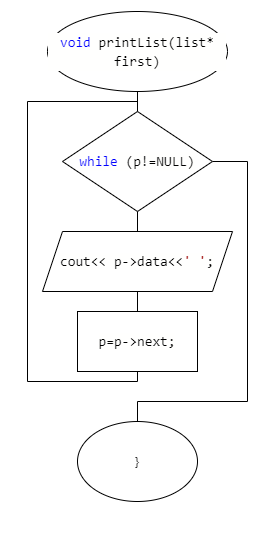
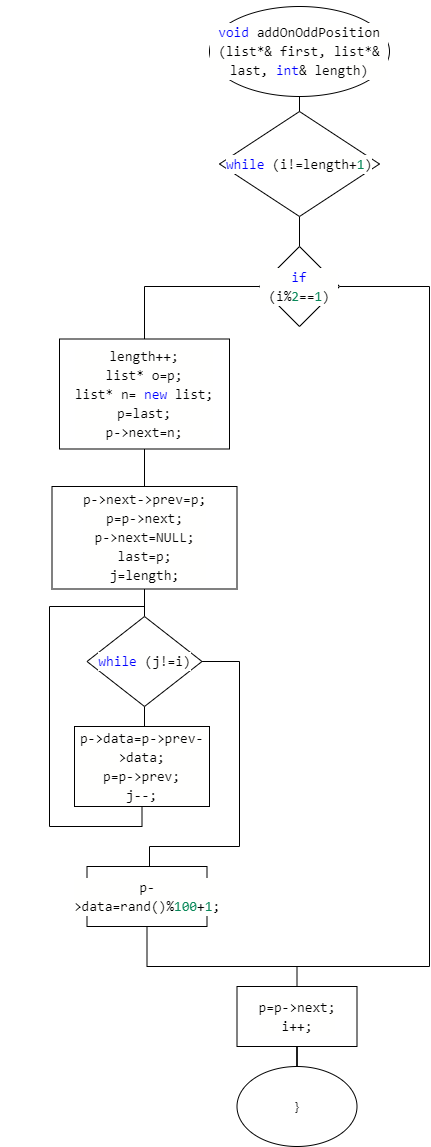
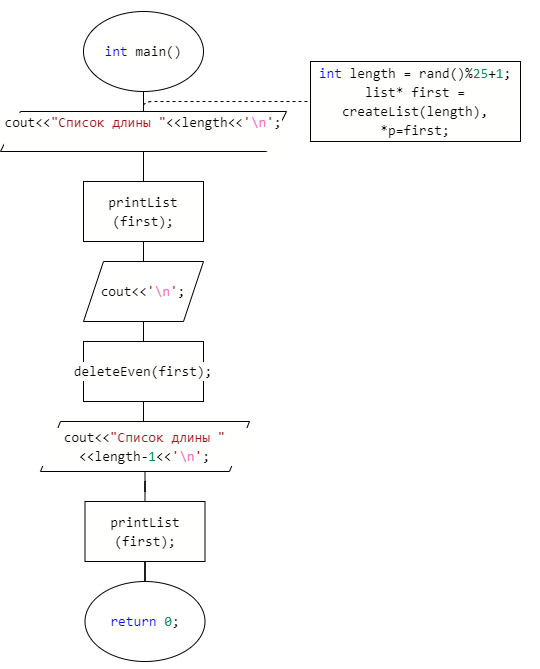
**Блок-схема программы (Стек, указатели)**

****

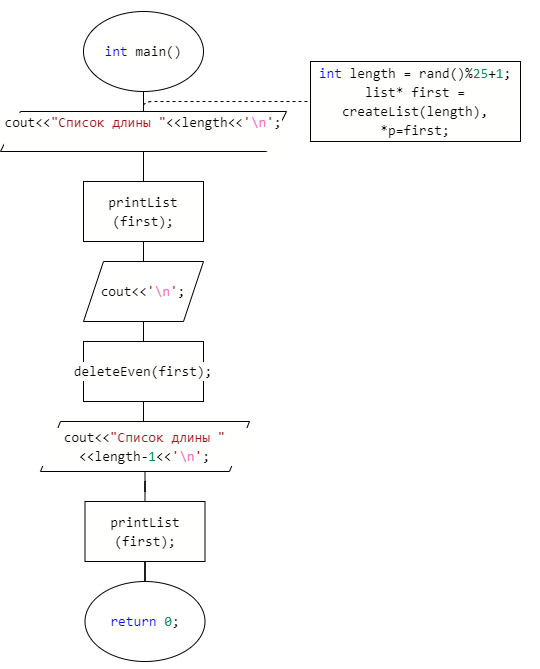
**Блок-схема программы (Стек, STL)**

****

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

****

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

****

**Решение (Однонаправленный список)**

#include <iostream>

using namespace std;

struct list {

int data;

list\* next;

};

list\* createList (int);

void printList (list\*);

void deleteEven (list\*);

int main()

{

int length = rand()%25+1;

list\* first = createList(length), \*p=first;

cout<<"Список длины "<<length<<'\n';

printList (first);

cout<<'\n';

deleteEven(first);

cout<<"Список длины "<<length-1<<'\n';

printList (first);

return 0;

}

list\* createList (int length)

{

list\* first, \*p;

int i;

first=NULL;

p=new list;

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

first=p;

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

{

list\* n = new list;

p->next = n;

p=p->next;

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

p->next=NULL;

}

return first;

}

void printList(list\* first)

{

list\* p = first;

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;

}

if (p->next==NULL)

{

cout<<"Не найдено четного элемента\n";

}

}

**Решение (Двунаправленный список, указатели)**

#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<<"Список 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 (Двунаправленный список, STL)**

#include <iostream>

#include <list>

#include <iterator>

#include <algorithm>

using namespace std;

int main()

{

list <int> list1;

list <int> list2;

int i,h;

for (i=0;i<10;i++)

{

list1.push\_back (rand()%25);

}

cout<<"Ваш список: ";

copy(list1.begin(),list1.end(), ostream\_iterator <int> (cout," "));

cout<<'\n';

for (i=0;i<10;i++)

{

list2.push\_back (rand()%25);

list2.push\_back (list1.front());

h=list1.front();

remove(list1.begin(),list1.end(),h);

}

cout<<"Ваш список: ";

copy(list2.begin(),list2.end(), ostream\_iterator <int> (cout," "));

cout<<'\n';

}

**Решение (Стек, указатели)**

//Удалить из стека первый элемент с четным информационным полем.

#include <iostream>

using namespace std;

struct list {

int data;

list\* next;

};

list\* createList (int);

void printList (list\*, int);

void deleteEven (list\*, int&);

int main()

{

int length = rand()%25+1;

list\* first = createList(length);

printList (first, length);

cout<<'\n';

deleteEven(first, length);

printList (first, length);

return 0;

}

list\* createList (int length)

{

list\* first, \*p;

int i;

first=NULL;

p=new list;

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

first=p;

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

{

list\* n = new list;

p->next = n;

p=p->next;

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

p->next=NULL;

}

return first;

}

void printList(list\* first, int length)

{

list\* p = first;

cout<<"Стек длины "<<length<<'\n';

while (p!=NULL)

{

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

p=p->next;

}

}

void deleteEven (list\* first, int& length)

{

list\* p = first;

while (p->next!=NULL)

{

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

{

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

length--;

break;

}

p=p->next;

}

}

**Решение (Стек, STL)**

#include <iostream>

#include <stack>

#include <iterator>

using namespace std;

int main()

{

stack <int> st;

stack <int> stNew;

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

bool f=false;

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

{

st.push (rand()%100);

}

cout<<"Ваш стек: ";

for (i=length-1; i>-1; i--)

{

cout<<st.top()<<' ';

arr[i]=st.top();

st.pop();

}

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

{

if (arr[i]%2==0 && !f)

{

arr[i]=-1;

f=true;

}

}

cout<<'\n';

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

{

stNew.push (arr[i]);

}

cout<<"Ваш стек: ";

for (i=length-1; i>-1; i--)

{

if (stNew.top()!=-1) cout<<stNew.top()<<' ';

arr[i]=stNew.top();

stNew.pop();

}

}

**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();

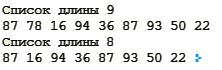
cout<<quNew.front()<<' ';

quNew.pop();

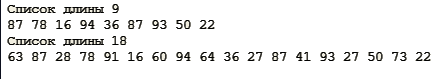
}

}

**Скриншоты результатов работы программы (Однонаправленный список)**

****

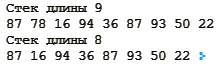
**Скриншоты результатов работы программы (Двунаправленный список, указатели)**

****

**Скриншоты результатов работы программы (Двунаправленный список, STL)**

****

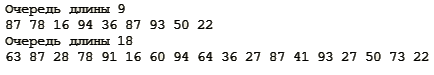
**Скриншоты результатов работы программы (Стек, указатели)**

****

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

****

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

****

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

****

**GitHub**

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