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 №11.3**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: Информационные и динамические структуры. Стеки

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 со стеками;

**Постановка задачи**

(Вариант 19)

1. Сформировать стек.

2. Распечатать полученную структуру.

3. Выполнить обработку структуры в соответствии с заданием.

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

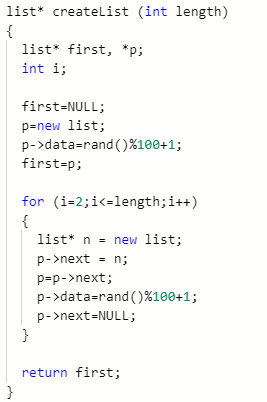
4. Распечатать полученный результат.

5. Удалить соответствующую структуру из памяти.

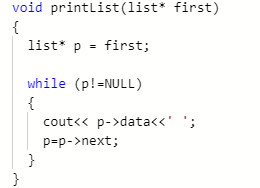
**Анализ задачи (Стек, Указатели)**

**1.** Для решения задачи необходимо:

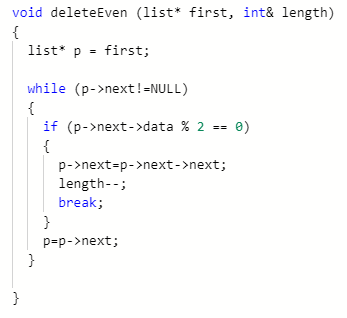
**1.1**. Реализовать функцию для создания стека



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

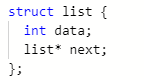


**1.3.** Реализовать функцию для удаления из стека первого элемента с четным информационным полем.



**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.

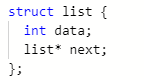


**2.2.** Для хранения адреса первого элемента стека используется тип list\*



**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.



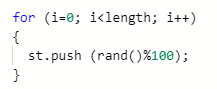
**Анализ задачи (Стек, STL)**

**1.** Для решения задачи необходимо:

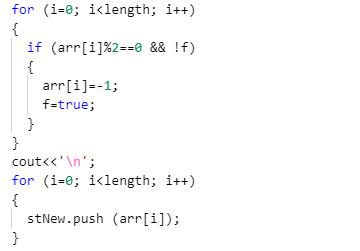
**1.1.** Для работы с двунаправленными списками необходимо подключить библиотеки stack и iterator.



**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.

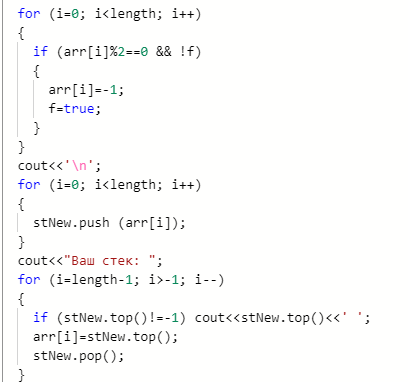


**4.2.** Для создания списка со случайными значениями использовалась функция rand().

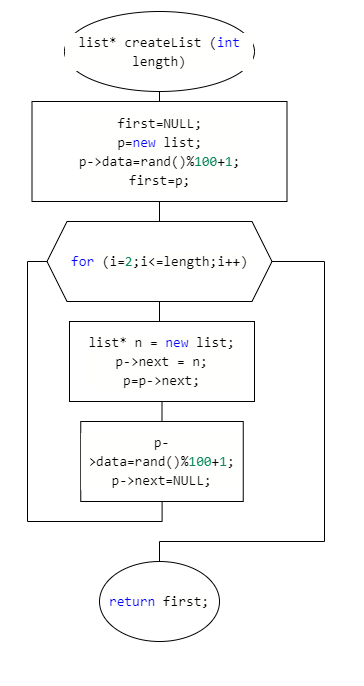
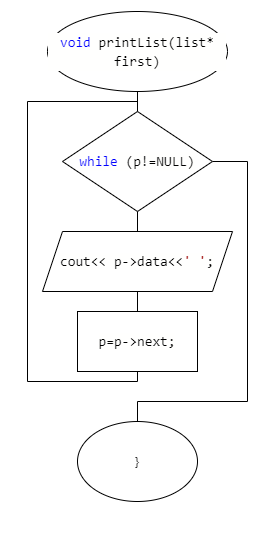
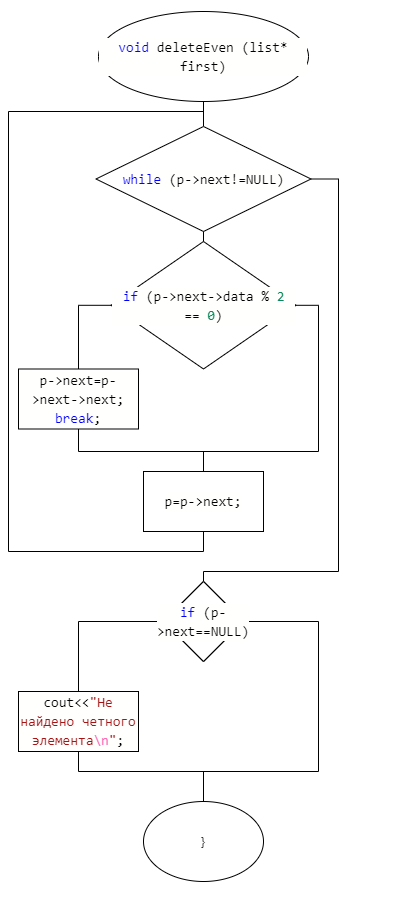
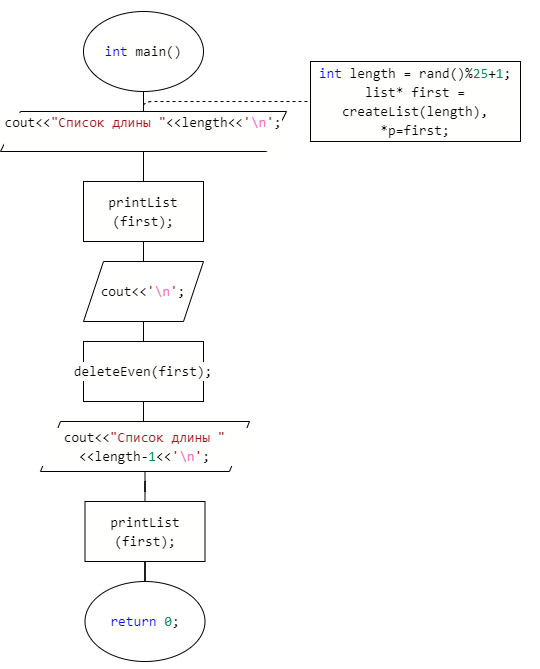


**5.** Поставленные задачи будут решены следующими действиями:

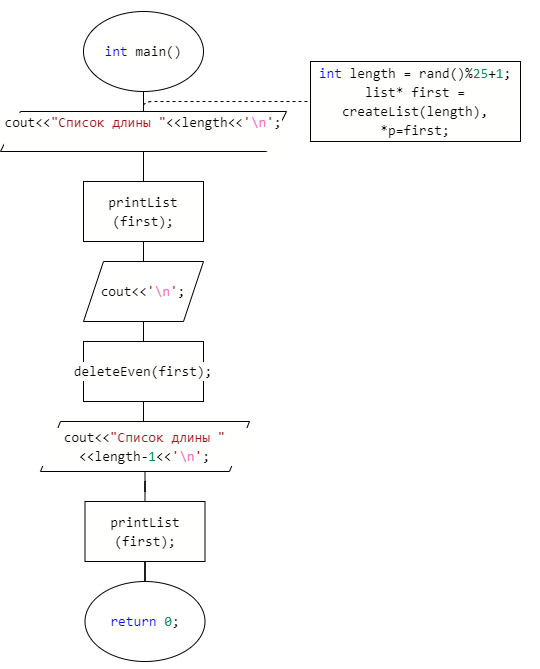
**5.1.** Будет применен код для удаления элемента с четным информационным полем, таким образом будет создан новый стек, который будет удовлетворять условиям задачи.



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

****

**Блок-схема программы (Стек, STL)**

****

**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()<<' ';

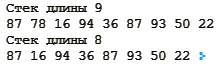
arr[i]=stNew.top();

stNew.pop();

}

}

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

****

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

****

**GitHub**

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