Лабораторная работа №5
Тема: Списки и стек.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Список – это совокупность объектов или элементов, в котором каждый объект содержит информацию о местоположении связанного с ним объекта.

Если список располагается в оперативной памяти, то, как правило, информация для поиска следующего объекта – это указатель, адрес памяти. Если связный список хранится на диске в файле, то информация о следующем элементе может включать смещение элемента от начала файла к положению указателя записи или считывания файла, ключ записи и любую другую информацию, позволяющую однозначно отыскать следующий элемент списка.

В списке элементы связаны друг с другом логически. Логический порядок следования элементов списка определяется с помощью указателей. Подчеркнем, что логический порядок следования элементов списка может не совпадать с физическим порядком их расположения в памяти ЭВМ.

Списки бывают линейными и кольцевыми, односвязными и двусвязными.

Как правило, элемент списка представляет собой структурную переменную, содержащую указатель или указатели на следующий элемент и любое число других полей, называемых информационными.

Если движение от элемента к элементу списка возможно только в одном направлении и список имеет начальную точку такого движения, говорят об односвязном списке. Элемент односвязного списка включает только указатель на следующий элемент. Сам список характеризуется указателем на начало списка (см. рисунок 1).

Двусвязный список позволяет выполнять «движение» от элемента к элементу в обоих направлениях. В этом случае элемент включает два указателя: на предыдущий и последующий элементы списка. А так как список имеет и начало, и конец, описываются еще два указателя – начала и конца списка (см. рисунок 2).

Список, в котором последний элемент не связан с первым, называется линейным. Соответственно, кольцевым называется список, у которого последний элемент указывает на первый.

В последнем элементе односвязного и двусвязного линейного списка указатель на следующий элемент равен нулю. В первом элементе двусвязного списка указатель на предыдущий элемент равен нулю.

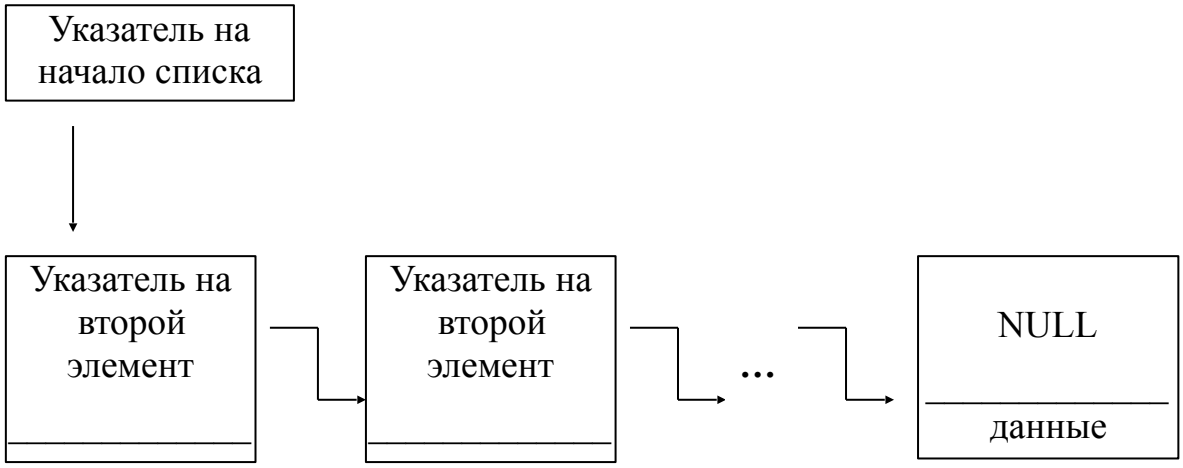

Рисунок 1 - Модель односвязного линейного списка

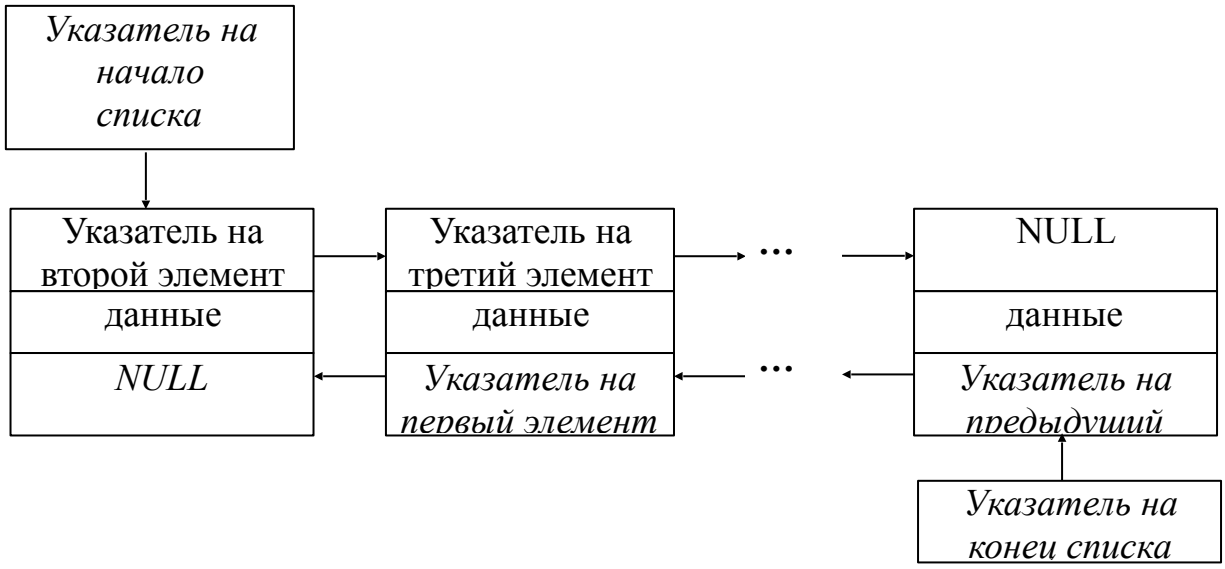

Рисунок 2 - Модель двусвязного линейного списка

2. Операции над списками.

Основными операциями, которые выполняются над списками, являются перебор элементов списка, поиск заданного элемента, вставка в список нового элемента, удаление элемента из списка. Выполнение этих операций основано на изменении указателей.

2.1. Перебор элементов списка.

Эта операция выполняется для линейных списков очень часто и состоит в последовательном доступе к элементам списка – ко всем до конца списка либо до нахождения искомого элемента. Для каждого из перебираемых элементов осуществляется некоторая обработка его информационной части:

сравнение с образцом, печать, модификация и прочее.

2.2. Вставка элемента в список.

Схематически выполнение этой операции представлено на рисунке 3.

Мы видим, что указатель элемента эл2 теперь указывает не на элемент эл3, а на элемент эл5. Указатель элемента эл5 указывает на элемент эл3. Логическая последовательность элементов будет эл1, эл2, эл5, эл3, эл4. Подчеркнем еще раз, что в списке новый элемент эл5 физически не обязательно помещается за элементом эл2. Достаточно, чтобы он следовал за ним логически.

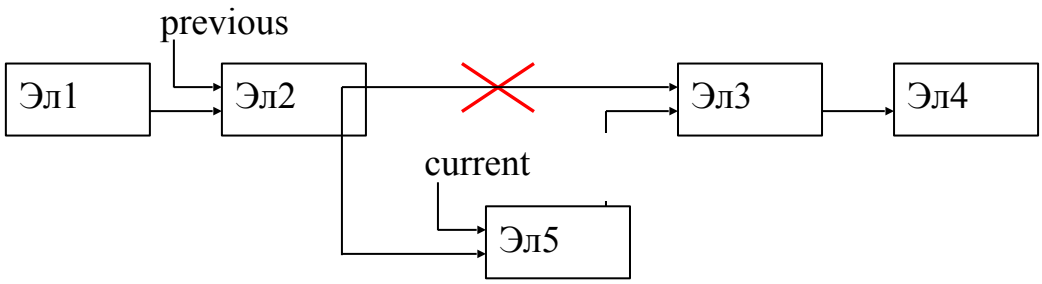

Рисунок 3 - Вставка элемента в середину односвязного списка

Вставка элемента в начало односвязного списка показана на рисунке 4. В данном случае переустанавливается указатель на начало списка на вновь вставленный элемент Эл3, а элемент Эл3 указывает на бывший первый элемент списка.

head – указатель на начало

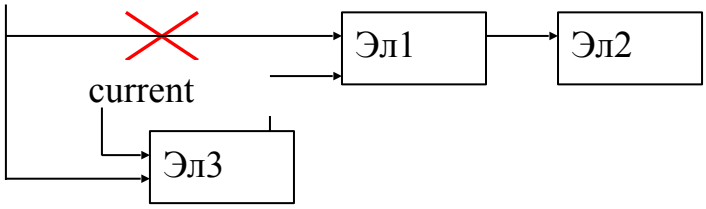

Рисунок 4 - Вставка элемента в начало односвязного списка

2.3. Удаление элемента из списка.

При удалении элемента Эл3 из списка, прежде всего, выполняется поиск этого элемента в списке, а затем его удаление. Результат удаления показан на рисунке 5.

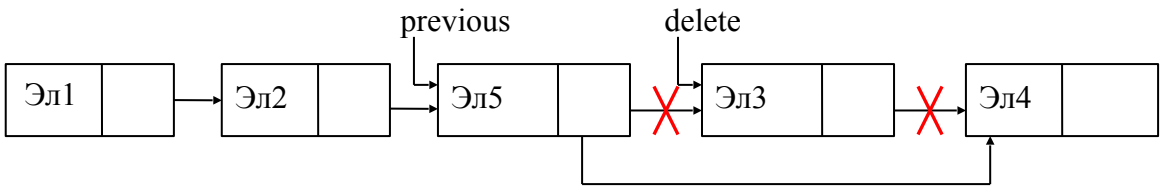

Рисунок 5 - Удаление элемента из списка

Теперь за элементом эл5 следует элемент эл4. Соответствующим образом изменился указатель элемента эл5. Элемент эл3 в списке теперь отсутствует, так как при просмотре списка, переходя от элемента к элементу в соответствии с указателями, мы на элемент эл3 не попадаем. Следует отметить, что удаление элемента из списка и удаление из памяти – это не одно и то же. Элемент Эл3 будет находиться в памяти до тех пор, пока мы не удалим его явно с помощью оператора delete.

Удаление элемента из начала списка показано на рисунке 6.

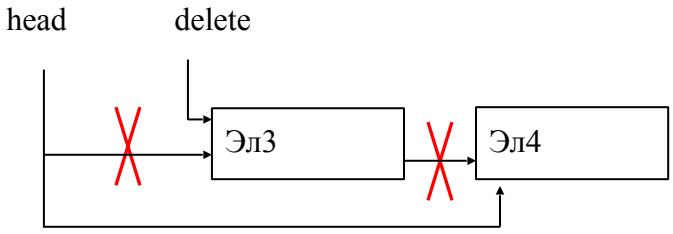

Рисунок 6 - Удаление элемента из начала списка

3. Пример реализации односвязного списка с помощью массива структур.

В рассматриваемом примере используется следующая структура

```
struct list
{
    char book;
    list *next;
};
```

Предполагается, что память, отводимая под элемент списка, выделяется динамически, поэтому при реализации списков его длина ограничивается только доступным объёмом памяти. Информационное поле представляет собой символьную переменную по имени book. Признаком последующего элемента списка является наличие поля next, а признаком последнего элемента списка является равенство на NULL этого поля.

Следующая программа реализует операции:

- создает пустой список;
- добавляет элементы в список в алфавитном порядке;
- удаляет элементы из списка;
- проверяет, является ли список пустым;
- выводит список на экран.

Функция *main* организует запуск программы *instructions* для ввода кода необходимой операции над списком, и вызывает функцию, выполняющую эту операцию.

Операцию добавления элемента в список в алфавитном порядке выполняет функция *insert*, операцию удаления – функция *del*, операцию проверки, является ли список пустым – *IsEmpty*, вывод списка на экран – *printList*.

```
#include "stdafx.h"
# include <stdio.h>
# include <stdlib.h>

struct list
{
    char book;
    struct list *next;
};
typedef struct list ListNode;
typedef ListNode *ListNodePtr;
void insert (ListNodePtr*, char);
char del (ListNodePtr*, char);
int IsEmpty (ListNodePtr);
void printList (ListNodePtr);
void instructions (void);

int main ()
{
    ListNodePtr start = NULL;
    int choice;
    char elem;
    instructions();                    // вывести меню
    printf("?");
    scanf("%d", &choice);
    while (choice !=3)
    {
    switch (choice)
    {
        case 1:             /*Добавить значение в список*/
        printf ("Enter an character: ");
```

```
        scanf("\n%c", &elem);
        insert(&start, elem);
        printList (start);
        break;
        case 2:             /*Удалить значение из списка*/
        if (!IsEmpty(start))
        {
            printf("Enter character to be deleted:");
            scanf("\n%c", &elem);
            if (del(&start,elem))
            {      printf("%c deleted.\n", elem);
                printList(start);  }
            else
            printf("%c not found.\n", elem);
        }
        else
        printf("List is empty.\n");
        break;
        default:
        printf ("Invalid choice.\n");
        instructions();
        break;
    }
    printf("?");
    scanf("%d", &choice);
    }
    printf("End of run.\n");
    return 0;
}
/*Вывести инструкции*/
void instructions(void)
{
    printf("Enter choice:\n"
    "1  insert an element into the list.\n"
    "2  delete an element from the list.\n"
    "3  end program.\n");
}
/*Вставить в список новое значение в алфавитном порядке */
void insert (ListNodePtr *s, char value)
{
    ListNodePtr newP, previous, current;
    NewP = (ListNodePtr) malloc (sizeof(ListNode));
    if (newP!=NULL)
    {     /*есть ли место ?*/
        newP->book =value;
        newP->next=NULL;
        previous=NULL;
```

```
        current=*s;
        while (current!=NULL && value >current->book)
        {     previous=current;
               current=current->next;    }
        if(previous==NULL)
        {      newP->next=*s;
            *s=newP;            }
        else
        {      previous->next = newP;
                newP->next = current;     }
    }
    else
    printf("%c not inserted. No memory available.\n", value);
}
/*Удалить элемент списка*/
char del (ListNodePtr *s, char value)
{
    ListNodePtr previous, current, temp;
    if (value == (*s)->book)
    {
        temp=*s;
        *s=(*s)->next;  // отсоединить узел
        free(temp);      // освободить отсоединенный узел
        return value;
    }
    else
    {      previous=*s;
        current=(*s)->next;
        while (current!=NULL && current->book!= value)
        {
        previous=current;           /*перейти.......*/
        current=current->next;     /*...к следующему*/
        }
        if (current!=NULL)
        {      temp=current;
            previous->next=current->next;
            free(temp);
            return value;    }
    }
    return '\0';
}
/*Возвратить 1, если список пуст, 0 в противном случае*/
int IsEmpty(ListNodePtr s)
{
return s==NULL;
}
/*Распечатать список*/
```

```
void printList (ListNodePtr current)
{
    if (current==NULL)
    printf("The list is empty.\n");
    else
    {      printf("The list is :\n");
        while (current!=NULL)
        {       printf("%c-->",current->book);
                 current=current->next;    }
    printf("NULL\n");
    }
}
```

Стек – это последовательный список переменной длины, включение и исключение элементов из которого выполняются только с одной стороны списка, называемого **вершиной** стека. Другие названия стека – «магазин» и очередь, функционирующая по принципу LIFO (Last – In – First – Out – "последним пришел – первым исключается"). Примеры стека: винтовочный патронный магазин, тупиковый железнодорожный разъезд для сортировки вагонов.

Основные операции над стеком – включение нового элемента (англ. push – заталкивать) и исключение элемента из стека (англ. pop – выскакивать).

Полезными могут быть также вспомогательные операции:

- определение текущего числа элементов в стеке;
- очистка стека;
- неразрушающее чтение элемента из вершины стека, которое может быть реализовано как комбинация основных операций:

```
x=pop(stack0);   push(stack0,x)
```

Для наглядности рассмотрим небольшой пример, демонстрирующий принцип включения элементов в стек и исключения элементов из стека. На рисунке 7 изображены следующие состояния стека:

- а) пустого;
- б–г) после последовательного включения в него элементов с именами 'A', 'B', 'C';
- д, е) после последовательного удаления из стека элементов 'C' и 'B';
- ж) после включения в стек элемента 'D'.

Рисунок 7 - Включение и исключение элементов из стека

Стек можно представить, например, в виде стопки книг (элементов), лежащей на столе. Присвоим каждой книге свое название, например A, B, C, D... Тогда в момент времени, когда на столе книги отсутствуют, про стек по аналогии можно сказать, что он пуст, т.е. не содержит ни одного элемента. Если же начинать последовательно класть книги одну на другую, то получим стопку книг (допустим из n книг), или получим стек, в котором содержится n элементов, причем вершиной его будет являться элемент n+1. Удаление элементов из стека осуществляется аналогичным образом, т. е. удаляется последовательно по одному элементу, начиная с вершины, или по одной книге из стопки.

Возможны различные варианты реализации стека, например на основе массива или односвязного списка. В случае реализации на основе односвязного списка минимальными необходимыми функциями является добавление элемента в начало списка и удаление с возвратом значения первого элемента.

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Варианты заданий

Задача 1 - максимум 4 балла при сдаче на том же занятии, на котором задача выдана. При сдаче на следующих занятиях максимум - 2 балла.

Задача 2 - максимум 4-5 баллов при сдаче на любом занятии.

Задача 1 должна быть сдана обязательно.

Вариант	Задание
1	1. Реализовать двусвязный список целочисленных значений. Предусмотреть возможность добавления элемента в конец списка, удаление элемента с конца списка, вывод всех элементов в обратном порядке. Позволить вводить и выводить элементы с клавиатуры. 2. Реализовать стек целочисленных значений на основе односвязного списка, функции push и pop. Добавить возможность удаления N-го элемента из стека при помощи только этих функций и дополнительного стека, N задать с клавиатуры.
2	1. Реализовать ввод текста по абзацам при помощи односвязного списка, элементы которого хранят строки. С клавиатуры задать размер отступа для абзаца (в кол-ве пробелов). При выводе текста для каждого абзаца применить заданный отступ. Добавить возможность добавления текста абзаца с клавиатуры перед N-м абзацем. 2. Реализовать стек целочисленных значений на основе односвязного списка, функции push и pop. При помощи только этих функций и дополнительного стека удалить из исходного стека все элементы, которые являются суммой двух соседних элементов. Повторять до тех пор, пока такие элементы встречаются после каждого удаления.
3	1. Реализовать односвязный список целочисленных значений, поддерживающий порядок по мере добавления элементов. Добавить возможность вывода элементов списка. 2. Реализовать стек целочисленных значений на основе односвязного списка, функции push и pop. Поменять местами N-й и M-й элементы стека, используя только эти функции. N и M задать с клавиатуры.

4	1. Реализовать односвязный список целочисленных значений, функцию добавления элемента списка в конец. Добавить функцию удаления N последних элементов списка. N ввести с клавиатуры. 2. При помощи стека определить, является ли правильной скобочная последовательность, введенная с клавиатуры. Предусмотреть работу с несколькими видами скобок: (), {}, []. Примеры. (()()(()) - неправильная, не хватает закрывающей скобки (()()(())).)(- неправильная, закрывающая скобка предшествует открывающей. ({)} - неправильный порядок закрытия скобок. ({()}())[] - правильная.
5	1. Реализовать двусвязный список символов с возможностью добавления в начало, добавления в конец. Добавить возможность вызвать каждый из вариантов добавления элементов. С клавиатуры ввести два числа: M и N. Поменять связь между элементами: разместить N-ый элемент перед M-ым. Дополнительных коллекций не использовать. 2. Реализовать стек целочисленных значений на основе односвязного списка, функции push и pop. Сделать циклический сдвиг вправо на N элементов, используя только эти функции и дополнительные стеки. Пример. Исходный стек: A->B->C->D, N = 2, результат: C->D->A->B.
6	1. Реализовать односвязный список целочисленных значений, функцию добавления и вывода элементов списка. По запросу с клавиатуры удалить средний элемент списка (два средних, если элементов четное количество). 2. Ввести строку. При помощи стека развернуть порядок слов в строке.
7	1. Реализовать односвязный список целочисленных значений с функцией добавления элемента в середину (слева от середины, если количество элементов нечетное), удаления первого элемента и вывода списка по запросу с клавиатуры. 2. Реализовать стек целочисленных значений на основе односвязного списка, функции push и pop. Отсортировать стек при помощи только этих функций и дополнительного стека.
8	1. Реализовать двусвязный список, элементы которого содержат цифру. Реализовать функцию добавления элемента в конец, вывода элементов. Сделать возможность ввести массив чисел, каждое из которых представлено списком цифр. Вывести все числа. 2. Реализовать стек символов на основе односвязного списка, функции push и pop. Ввести значения стека с клавиатуры. При помощи дополнительного стека удалить из исходного стека повторяющиеся значения. Других дополнительных коллекций не использовать.

9	1. Реализовать односвязный список целочисленных значений, функцию добавления элемента списка в конец. С клавиатуры ввести N. Удалить N-й с конца элемент. 2. Реализовать стек целочисленных значений на основе односвязного списка, функции push и pop. При помощи дополнительных стеков переместить все четные элементы в конец исходного стека, сохраняя порядок остальных элементов.
10	1. Реализовать односвязный список целочисленных значений. Сделать возможность добавления элементов и их вывода на экран. С клавиатуры ввести число N. Добавить это число после элемента, значение которого меньше, чем это число и перед элементом, значение которого больше, чем это число. В случае, если для всего списка это условие не выполняется, вывести соответствующее сообщение на экран. 2. Реализовать стек и двусвязный список, элементы которых содержат цифру. Сделать возможность ввести два числа, каждое из которых представлено двусвязным списком цифр. Поразрядно сложить эти числа, помещая результат сложения каждого разряда в стек. Вывести результат.
11	1. Реализовать двусвязный список значений с плавающей запятой. Ввести элементы списка. С клавиатуры ввести индекс N. Переместить N-й элемент списка в конец списка. 2. Реализовать стек символов на основе односвязного списка, функции push и pop. Ввести число M и символы с клавиатуры. С помощью стека проверить, является ли введенная строка симметричной относительно символа с индексом M без учета пробелов, точек и запятых. Пример: строка “а б вг, д еж е, дг. вба”, M = 11 (символ “ж"), результат - строка симметрична. Дополнительных коллекций не использовать, исходную строку обрабатывать посимвольно.
12	1. Реализовать односвязный список целочисленных значений с функцией добавления элемента в конец. Сделать возможность с клавиатуры добавить элемент, просмотреть все элементы. С клавиатуры ввести два числа: M и N. Поменять связь между элементами: разместить N-й элемент после M-го. 2. Реализовать эмуляцию текстового редактора с двумя функциями: добавить текст в конец, удалить текст между двумя индексами. Реализовать undo стек, который позволит отменять любое количество этих операций.

13	1. Реализовать двусвязный список целочисленных значений. С клавиатуры задать максимальный размер списка. При добавлении элемента проверять, будет ли превышен размер списка. Если будет, то добавить элемент и удалить элемент с минимальными значением. 2. Реализовать мини-игру. До начала задается интервал от M до N и время хода T. Случайным образом генерируется стек из 10 чисел от M до N. На каждый ход из этих 10 чисел берется число, квадрат которого выводится на экран. Нужно угадать, какое число было возведено в квадрат. Если ответ неправильный или на ответ ушло времени больше, чем T, то число помещается в стек неправильных ответов. Если ответ правильный, то у игрока появляется возможность дополнительно ответить на вопрос с вершины стека неправильных ответов. Игрок побеждает, если к моменту окончания исходных чисел в стеке неправильных ответов нет ни одного числа.
14	1. Реализовать односвязный список целочисленных значений с функцией добавления элемента в конец. По запросу с клавиатуры удалить из списка все элементы с повторяющимися значениями. 2. Реализовать стек строк на основе односвязного списка, функции push и pop. Сделать простой эмулятор перехода по страницам браузера: возможность ввести адрес страницы, вернуться на предыдущую (вплоть до первой) и следующую страницу (кнопки назад/вперед в браузере). Необходимо реализовать только отображение текущего адреса, загружать страницу не нужно.

15	1. Реализовать односвязный список целочисленных значений с функцией добавления элемента в конец, удаления последнего элемента. При добавлении элемента в конец автоматически добавляется такой же элемент в начало. При удалении последнего элемента автоматически удаляется первый. Реализовать возможность вывода всех элементов списка на экран. 2. Дана структура, представляющая элемент одежды и содержащая уникальное название и количество использований. Реализовать стек элементов одежды на основе односвязного списка, функции push и pop. Добавить стек чистых вещей и список вещей, нуждающихся в стирке. Каждая должна иметь уникальный идентификатор. Добавить возможность надеть верхнюю вещь из стека чистых вещей (при этом увеличивается счетчик использования вещи), отправить надетую вещь в список для стирки, постирать весь список с отправкой постиранных вещей в стек чистых вещей. Сделать возможность вывода всей информации о чистых вещах и вещах в списке для стирки.