
Front matter

title: "Лабораторная работа н.1" subtitle: "Односвязный список" author: "Петров Артем Евгеньевич"

Generic options

lang: ru-RU toc-title: "Содержание"

Bibliography

bibliography: bib/cite.bib csl: pandoc/csl/gost-r-7-0-5-2008-numeric.csl

Pdf output format

toc: true # Table of contents toc-depth: 2 lof: true # List of figures lot: true # List of tables fontsize: 12pt
linestretch: 1.5 papersize: a4 documentclass: scrreprt

l18n polyglossia

polyglossia-lang: name: russian options: - spelling=modern - babelshorthands=true polyglossia-otherlangs:
name: english

l18n babel

babel-lang: russian babel-otherlangs: english

Fonts

mainfont: PT Serif romanfont: PT Serif sansfont: PT Sans monofont: PT Mono mainfontoptions: Ligatures=TeX
romanfontoptions: Ligatures=TeX sansfontoptions: Ligatures=TeX,Scale=MatchLowercase monofontoptions:
Scale=MatchLowercase,Scale=0.9

Biblatex

biblatex: true biblio-style: "gost-numeric" biblatexoptions:

- parenttracker=true
- backend=biber
- hyperref=auto
- language=auto
- autolang=other*
- citestyle=gost-numeric

Pandoc-crossref LaTeX customization

figureTitle: "Рис." tableTitle: "Таблица" listingTitle: "Листинг" lofTitle: "Список иллюстраций" lotTitle:
"Список таблиц" lolTitle: "Листинги"

Misc options

indent: true header-includes:

- `\usepackage{indentfirst}`
- `\usepackage{float} # keep figures where there are in the text`
- `\floatplacement{figure}{H} # keep figures where there are in the text`

Цель работы

Целью данной работы служит идея научиться работать с классами и структурами в банальнейшем их представлении-односвязном списке, дополненном различными функциями: добавление элемента в начало списка, в конец, перед выбранным элементом, после выбранного элемента, удаление узла по его имени и печать всего списка.

Задание

Написать компьютерную программу, содержащую

- Описание структуры, содержащей поля типа `string`, `int`, `double`;
- Набор функций для работы со списком на базе этой структуры:
 - Добавление элемента в начало списка;
 - Добавление элемента в конец списка;
 - Добавление элемента в список после заданного элемента;
 - Добавление элемента в список перед заданным элементом;
 - Удаление из списка элемента с заданным именем;
 - Вывод содержания списка на экран;
- Функцию `main`, содержащую сценарий работы со списком, использующий разработанный инструментарий.

Выполнение лабораторной работы

ОТЧЕТ ПОЛНОСТЬЮ НАПИСАН НА ЯЗЫКЕ РАЗМЕТКИ MARKDOWN!!!

Задание 1. Создание структуры списка.

1. При создании односвязного списка я руководствовался следующей идеей:

Создать две структуры-структуру Звена(Node, далее Thing) и структуру списка, чтобы в дальнейшем структура Списка, со всеми добавлениями, оставалась полной до завершения программы. Кроме того, структура самого бы списка содержала все методы класса(или структуры) для достижения полной автономности внутри себя, что позволило бы использовать обе структуры в любой другой

функции(помимо main), любом другом классе и т.д. Таким образом, программа становилась бы невероятно гибкой для дополнения, обновления, внедрения и т.д. В итоге, я так же смог добиться небольшого пользовательского интерфейса, который при использовании обывателем не требует знания языка программирования.

2. Приступим же к описанию структур:

1. Структура звена, которое в нашем случае является единицей товара в магазине или буфете. Следовательно, звено должно содержать сведения о своем названии, цене, количестве в магазине, на складе и т.п. Однако, т.к. все реализуется на примере списка, надо так же добавить указатель на адрес в памяти на следующий элемент. Выглядеть это будет так:

```
struct Thing
{
    string name; //имя
    float price; //цена(значение с плавающей запятой при надобности)
    int amount; //количество
    Thing* next; // указатель на адрес след. элемента
};
```

2. Структура списка элементов, которая будет содержать методы, указанные в задании, и которая обязана содержать указатель на первый(начальный) элемент списка(англ. head). В начале она будет выглядеть так:

```
struct List
{
    Thing* head=NULL; // указатель на первый элемент(по счету)
    /* тут будут различные методы структуры */
};
```

Задание 2. Создание набора функций на базе структуры Thing в List

1. СОЗДАНИЕ НЕЗАВИСИМОГО ЗВЕНА. Для начала, стоит сказать, что создание самого элемента будет нужно нам минимум для 4 функций из заданных 6, поэтому если добавлять пользовательский интерфейс в терминал(или имитировать его), то огромного количества команд ввода и считываний значений не избежать, если не сделать создание звена без заведомого включения в список. Поэтому(внутри структуры List):

```
Thing* createNode()//обязательно делаем функцию того же типа, что и звено,
//чтобы в будущем возвращать адрес новосозданной ячейки. Указатель нужен для того,
чтобы не
// плодить сущности, ведь по сути ниже tovar является указателем типа Thing,
который
//получает адрес памяти, зарезервированной командой new под тип Звена(Thing).
{
    //ввод названия от пользователя
```

```

cout<< "-----" << endl;
cout << "\nVvedite nazvanie novogo producta: ";
string nazvanie;
cin >> nazvanie;

//Ввод цены для нового продукта
cout << "\nVvedite tsenu novogo producta: ";
float tsena;
cin >> tsena;

//Ввод количества для нового пользователя
cout << "\nVvedite kolichestvo novogo tovara: ";
int kolichestvo;
cin >> kolichestvo;

Thing *tovar=new Thing; //Выделение памяти под тип данных Thing(т.е.
// под все его параметры)
// Теперь адресам в памяти нового звена присвоим введенные значения
// при помощи указателя нового звена на его параметры
tovar->amount=kolichestvo;
tovar->name=nazvanie;
tovar->price=tsena;
tovar->next=NULL; //зануляем указатель на след. элемент, чтобы
//избежать непредвиденных значений
return tovar; //возвращаем эту ячейку, чтобы в будущем эта функция
// передавала значение в другие функции
}

```

Теперь приступим к написанию заданных функций.

2. ДОБАВЛЕНИЕ ЭЛЕМЕНТА В НАЧАЛО СПИСКА. Будем рассматривать два случая: Головного элемента нет и он уже есть. В случае, если его нет, мы присваиваем значению head из поля списка значение нового звена, созданного функцией createNode. В ином же случае, мы сначала создаем временную переменную(TMP), которой присваиваем значение прошлого головного элемента, потом присваиваем переменной головного элемента новое звено, а адресом следующего элемента послужит уже переменная TMP. Т.е. head->next=TMP. Выглядеть будет так:

```

void PushToHead(Thing* node) // Передаем в функцию при вызове значение нового
звена
{
    Thing* TMP=new Thing;
    TMP=head;
    if(head==NULL)
    {
        head=node;
        head->next=NULL;
    }
    else
    {
        head=node;
        head->next=TMP;
    }
}

```

```

    }
}

```

3. ДОБАВЛЕНИЕ ЭЛЕМЕНТА В КОНЕЦ СПИСКА. Рассмотрим три случая: Если головной элемент отсутствует, если отсутствует элемент следующий за головным элементом и если ни одно из этих условий не выполняется. В первом случае мы присваиваем головному элементу наш новый элемент. Во втором мы указателю головного элемента на следующий элемент присваиваем адрес нового элемента. В третьем же мы опять создаем переменную tmp3, которой присваиваем головной элемент и циклом прокручиваем нашу переменную, пошагово присваивая ей ее же указатель на следующий элемент, пока указатель переменной не будет указывать на NULL, потом же присваиваем указателю переменной на след. элемент наше новое звено.

```

void PushToTail(Thing* node)
{
    if(head==NULL)//Первый случай
    {
        head=node;
        node->next=NULL;
    }
    else if (head->next == NULL)//Второй случай
    {
        head->next=node;
        node->next=NULL;
    }
    else // Третий случай
    {
        Thing* tmp3=head;
        while (tmp3->next != NULL)
        {
            tmp3=tmp3->next;
        }
        tmp3->next=node;
    }
}

```

4. ДОБАВЛЕНИЕ ЭЛЕМЕНТ ПЕРЕД ЗАДАНЫМ. В этом случае мы создадим две переменные структуры звена-предыдущий элемент и ведущий, чтобы вклинить между ними элемент.

```

int AddBefore(Thing* node, unsigned int BeforeWhatElem) /*интовое значение,
которое означает, перед каким элементов по счету добавлять(позже мы введем функцию
печати, где будет отображен порядковый номер каждого)*/
{
    Thing* tmp4=new Thing;
    Thing* prev=new Thing;
    prev = NULL;
    tmp4=head;
    unsigned int counter1=0; /*счетчик, который в цикле будет инкрементироваться,
пока не

```

```

        сравнивается с номером того элемента, перед которым надо добавить.

        if (BeforeWhatElem == 0) //случай, если надо добавить перед головным, сделав
        новый эл.
        //головным
        {
            head=node;
            head->next=tmp4;
            return 0;
        }

        while (counter1 != BeforeWhatElem) //цикл, итерирующийся, пока текущим
        элементом не
        //станет тот, перед которым надо вставить звено
        {
            if (tmp4 == NULL) //если цикл прокрутился до значения текущего эл NULL, то
            остановка
            {
                cout << "no such elem";
                return 0;
            }
            prev=tmp4; //сохраняем предыдущий текущий эл от прошлой итерации, чтобы
            вставить
            //между ними без потери предыдущего
            tmp4=tmp4->next;
            counter1++;
        }
        prev->next=node; //делаем указателем на следующий эл адрес нового эл и
        перебрасываем
        //указатель нового эл на след. на текущий(тот перед которым вставляем)
        node->next=tmp4;
        return 0;
    }

```

5. ДОБАВЛЕНИЕ ЭЛЕМЕНТА ПОСЛЕ ЗАДАННОГО. На самом деле, мы повторяем предыдущую функцию, но цикл проделывает на шаг больше, делай уже тот элемент, после которого надо добавить, предыдущим(для цикла)

```

int AddAfter(Thing* node, int AddAfterWhat)
{
    Thing* tmp5=new Thing;
    Thing* prev1=new Thing;
    prev1 = NULL;
    tmp5=head;
    unsigned int counter1=0;

    if (AddAfterWhat == 0)
    {
        head=node;
        head->next=tmp5;
        return 0;
    }

```

```

    }

    while (counter1 <= AddAfterWhat)
    {
        if (tmp5 == NULL)
        {
            cout << "no such elem";
            return 0;
        }
        prev1=tmp5;
        tmp5=tmp5->next;
        counter1++;
    }
    prev1->next=node;
    node->next=tmp5;
    return 0;
}

```

6. УДАЛЕНИЕ ЭЛЕМЕНТА ИЗ СПИСКА С ЗАДАННЫМ ИМЕНЕМ. В данном случае мы повторим немного две предыдущие функции, т.к. нам придется связать два элемента между собой после удаления одного.

```

int deleteNode()
{
    cout << "\nVvedite nazvanie tovara, kotoryy nado udalit': ";
    string imya;
    cin >> imya;

    Thing* prev=new Thing;
    prev=NULL;
    Thing* tmp3=new Thing;
    tmp3=head;

    if (tmp3->name == imya) //если головной элемент является нужным
    {
        delete tmp3;
        head=head->next; //следующий за головным становится головным
        return 0;
    }

    while (tmp3->name != imya)
    {
        prev=tmp3;
        if (tmp3 == NULL) //если прошли весь список, не найдя элемента
        {
            cout << "\nNet takogo elementa";
            return 0;
        }
        tmp3=tmp3->next;
    }
}

```

```

    tmp3=tmp3->next; //делаем еще один шаг по списку, а пред. остается на два сзади.
    prev->next=tmp3; //привязываем предыдущий к тому, что на два шага впереди.
    //возможно, я захламляю память, но я иначе создавал бы очередную бы
переменную.
    return 0;
}

```

7. ПЕЧАТЬ ВСЕГО СПИСКА

```

void printAll()
{
    Thing* tmp=new Thing; //создаем переменную для прошагивания списка циклом
    tmp=head; //копируем в нее голову.
    unsigned int counter=0; //очевидно, счетчик
    cout << setiosflags(ios::left) << setw(5) << "Num" << setw(10) << "Name" <<
    setw(10) << "Price" << setw(10) << "Amount" << endl;
    while(tmp != NULL) //печать циклом, пока текущий элемент не выйдет за пределы
списка
    {
        cout << setiosflags(ios::left) << setw(5) << counter << setw(10) << tmp-
>name << setw(10) << tmp->price << setw(10) << tmp->amount << endl;
        tmp=tmp->next;
        counter++;
    }
    delete tmp; //удаляем элемент
}

```

8. ИМИТИРОВАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА. Выполним ее оператором ветвления switch. Всего у нас 6 функций-методов списка, добавим еще выход, полное завершение программы. На самом деле, можно спокойно обойтись и без нее, но создавать новые списки придется с помощью программного кода. Кроме того, используется функция прочтения символа getch() (conio.h) и рекурсия для повторения выбора функции. Позже я приложу два варианта, когда будет рассматриваться функция main.

```

int ViborFunktсии()
{
    cout << "Spisok funktsiy:\n" << "1. Create & Push to head\n" << "2. Create &
Push to tail\n" << "3. Print all\n" << "4. Delete node\n" << "5. Create & push
before smth\n" << "6. Create & push after smth\n" << "7. quit\n";
    int choice;
    cin >> choice;
    switch (choice)
    {
    case 1:
        PushToHead(createNode());
        cout << "\nContinue?(y/n):";
        if (getch() == 'y')
        {
            ViborFunktсии();
        }
    }
}

```



```
        break;
    }
    else
    {
        break;
    }
case 2:
    PushToTail(createNode());
    cout << "Continue?(y/n):";
    if (getch() == 'y')
    {
        ViborFunktsii();
        break;
    }
    else
    {
        break;
    }
case 3:
    printAll();
    cout << "Continue?(y/n):";
    if (getch() == 'y')
    {
        ViborFunktsii();
        break;
    }
    else
    {
        break;
    }
case 4:
    deleteNode();
    cout << "Continue?(y/n):";
    if (getch() == 'y')
    {
        ViborFunktsii();
        break;
    }
    else
    {
        break;
    }
case 5:
    cout << "Vvedite poryadkoviy nomer:";
    int a;
    cin >> a;
    AddAfter(createNode(), a);
    cout << "Continue?(y/n):";
    if (getch() == 'y')
    {
        ViborFunktsii();
        break;
    }
    else
```

```

        {
            break;
        }
    case 6:
        cout << "Vvedite poryadkoviy nomer:";
        int b;
        cin >> b;
        AddBefore(createNode(), b);
        cout << "Continue?(y/n):";
        if (getch() == 'y')
        {
            ViborFunktsii();
            break;
        }
        else
        {
            break;
        }
    case 7:
        break;
    default:
        cout << "Net takogo varianta!!!";
        ViborFunktsii();
        break;
}
}

```

- Полный код будет в приложенном файле!!! Не забудьте, что все вышеописанные функции являются методом структуры list!!!

Задание 3. Функция main, содержащая сценарий работы со списком, использующая разработанный инструментарий.

1. Вариант

```

int main()
{
    List* lst=new List;
    lst->PushToHead(lst->createNode());
    lst->PushToHead(lst->createNode());
    lst->PushToTail(lst->createNode()); //В аргументах передаем метод создания
звена
    //В данном случае мы вызываем функцию через указатель на метод экземпляра
    //класса(ПО АДРЕСУ)
    lst->printAll();
    lst->deleteNode();
    lst->AddBefore(lst->createNode(), 0); //добавим перед первым(счет с 0
начинается)
    lst->printAll();
    lst->AddAfter(lst->createNode(), 1); //добавим после второго элемента
    lst->printAll();
}

```

```
    delete lst;  
    return 0;  
}
```

2. Вариант(с использованием функции н.8, иначе ее можно просто закомментировать и вернуться к первому варианту)

```
int main()  
{  
    cout << "\nDvusvyazniy spisok by Petrov Artem\n";  
    List* lst=new List; //создаем экземпляр структуры(класса)  
    lst->ViborFunktsii(); //указатель на адрес метода внутри структуры(класса)  
    delete lst; //удаляем, чтобы не захламлять память, я бы мог все привести к  
    полному  
    //интерфейсу с сохранением данного списка, но появилось много других дел)  
    return 0;  
}
```

- Пример выполнения функции(в двух вариантах):

1. См. рис. [-@fig:001]

```
Select H:\vscodevrsthwrd\damn\differentshit\1stlabmyown.exe
-----
Vvedite nazvanie novogo producta: e
Vvedite tsenu novogo producta: 1
Vvedite kolichestvo novogo tovara: 1
-----
Vvedite nazvanie novogo producta: r
Vvedite tsenu novogo producta: 2
Vvedite kolichestvo novogo tovara: 2
-----
Vvedite nazvanie novogo producta: t
Vvedite tsenu novogo producta: 3
Vvedite kolichestvo novogo tovara: 3
Num  Name      Price  Amount
0    r          2       2
1    e          1       1
2    t          3       3
Vvedite nazvanie tovara, kotoriy nado udalit':
-----
Vvedite nazvanie novogo producta: m
Vvedite tsenu novogo producta: 5
Vvedite kolichestvo novogo tovara: 5
Num  Name      Price  Amount
0    m          5       5
1    r          2       2
2    e          1       1
-----
Vvedite nazvanie novogo producta: q
Vvedite tsenu novogo producta: 6
Vvedite kolichestvo novogo tovara: 6
Num  Name      Price  Amount
0    m          5       5
1    r          2       2
2    q          6       6
3    e          1       1
```

{ #fig:001 width=70% }

2. См. рис. [-@fig:002]-[-@fig:004]

```
H:\vscodevrsthwrd\damn\differentshit\1stlabmyown.exe
Dvusvyazniy spisok by Petrov Artem
Spisok funktsiy:
1. Create & Push to head
2. Create & Push to tail
3. Print all
4. Delete node
5. Create & push after smth
```

```

5. Create & push after smth
6. Create & push before smth
7. quit
1
-----

Vvedite nazvanie novogo producta: e
Vvedite tsenu novogo producta: 1
Vvedite kolichestvo novogo tovara: 1
Continue?(y/n):Spisok funktsiy:
1. Create & Push to head
2. Create & Push to tail
3. Print all
4. Delete node
5. Create & push after smth
6. Create & push before smth
7. quit
2
-----

Vvedite nazvanie novogo producta: r
Vvedite tsenu novogo producta: 2
Vvedite kolichestvo novogo tovara: 2
Continue?(y/n):Spisok funktsiy:
1. Create & Push to head
2. Create & Push to tail
3. Print all
4. Delete node
5. Create & push after smth
6. Create & push before smth
7. quit
3
Num  Name      Price   Amount
0    e         1       1
1    r         2       2
Continue?(y/n):Spisok funktsiy:
1. Create & Push to head
2. Create & Push to tail
3. Print all
4. Delete node
5. Create & push after smth
6. Create & push before smth
7. quit
5
Vvedite poryadkoviy nomer:1
-----

Vvedite nazvanie novogo producta: t
Vvedite tsenu novogo producta: 3
Vvedite kolichestvo novogo tovara: 3
Continue?(y/n):Spisok funktsiy:
1. Create & Push to head

```

{ #fig:002 width=70% }

```

Select H:\vscodevrsthwrd\damn\differentshit\1stlabmyown.exe
Vvedite kolichestvo novogo tovara: 3
Continue?(y/n):Spisok funktsiy:
1. Create & Push to head

```

```
2. Create & Push to tail
3. Print all
4. Delete node
5. Create & push after smth
6. Create & push before smth
7. quit
5
Vvedite poryadkoviy nomer:0
-----

Vvedite nazvanie novogo producta: y

Vvedite tsenu novogo producta: 4

Vvedite kolichestvo novogo tovara: 4
Continue?(y/n):Spisok funktsiy:
1. Create & Push to head
2. Create & Push to tail
3. Print all
4. Delete node
5. Create & push after smth
6. Create & push before smth
7. quit
3
Num  Name      Price   Amount
0    e         1       1
1    y         4       4
2    r         2       2
3    t         3       3
Continue?(y/n):Spisok funktsiy:
1. Create & Push to head
2. Create & Push to tail
3. Print all
4. Delete node
5. Create & push after smth
6. Create & push before smth
7. quit
6
Vvedite poryadkoviy nomer:3
-----

Vvedite nazvanie novogo producta: u

Vvedite tsenu novogo producta: 5

Vvedite kolichestvo novogo tovara: 5
Continue?(y/n):Spisok funktsiy:
1. Create & Push to head
2. Create & Push to tail
3. Print all
4. Delete node
5. Create & push after smth
6. Create & push before smth
7. quit
3
Num  Name      Price   Amount
0    e         1       1
1    y         4       4
2    r         2       2
3    u         5       5
4    t         3       3
Continue?(y/n):Spisok funktsiy:
1. Create & Push to head
```

{ #fig:003 width=70% }

```

H:\vscodevrsthwrld\damn\differentshit\1stlabmyown.exe
Num  Name      Price  Amount
0    e         1      1
1    y         4      4
2    r         2      2
3    t         3      3
Continue?(y/n):Spisok funktsiy:
1. Create & Push to head
2. Create & Push to tail
3. Print all
4. Delete node
5. Create & push after smth
6. Create & push before smth
7. quit
6
Vvedite poryadkoviy nomer:3
-----

Vvedite nazvanie novogo producta: u

Vvedite tsenu novogo producta: 5

Vvedite kolichestvo novogo tovara: 5
Continue?(y/n):Spisok funktsiy:
1. Create & Push to head
2. Create & Push to tail
3. Print all
4. Delete node
5. Create & push after smth
6. Create & push before smth
7. quit
3
Num  Name      Price  Amount
0    e         1      1
1    y         4      4
2    r         2      2
3    u         5      5
4    t         3      3
Continue?(y/n):Spisok funktsiy:
1. Create & Push to head
2. Create & Push to tail
3. Print all
4. Delete node
5. Create & push after smth
6. Create & push before smth
7. quit
4

Vvedite nazvanie tovara, kotoriy nado udal
Continue?(y/n):Spisok funktsiy:
1. Create & Push to head
2. Create & Push to tail
3. Print all
4. Delete node
5. Create & push after smth
6. Create & push before smth
7. quit
3
Num  Name      Price  Amount
0    y         4      4
1    r         2      2
2    u         5      5
3    t         3      3
Continue?(y/n):

```



Выводы

В ходе данной лабораторной работы я научился работать с структурами, особенно с списками, т.к. двусвязный представляет упрощенный вариант. Кроме того, я попрактиковал понимание адресного пространства, методов структур, указателей и т.д. Так же я написал программу односвязного списка, который при небольшой корректировке мог бы быть использован с пользой.