Лабораторная работа № 1. Динамические структуры: стеки, очереди и списки

Цель работы: освоить программирование задач с динамическими структурами (стеки, очереди, списки)

Теоретическое обоснование

1. Стек

Кроме стандартных структур данных: список, множество, ассоциативный массив (словарь), в программировании используется и ряд других структур. Особенность каждой из них состоит в способе хранения данных и алгоритме доступа к элементам структуры. При использовании каждой структуры есть определённые ограничения - то, что структура «не умеет делать быстро». Например, операция добавления элемента в конец списка выполняется быстро, а удаление первого элемента выполняется за длину списка. Познакомимся с несколькими структурами, которые иногда называют «линейными структурами данных», потому что элементы в таких структурах данных хранятся в виде последовательности элементов, один за другим.

Например, список в языке Python и вектор в языке C++ также являются линейной структурой данных, у которой есть операция доступа к i-му по счёту элементу, и эта операция выполняется быстро. А вот такие операции, как удаление элементов из начала или вставка элемента в середину списка или вектора, выполняются долго.

Стек (stack) - это линейная структура данных, в которой элементы добавляются и удаляются только с одного конца - вершины стека. Можно представить себе стек как стопку тарелок. Вы можете положить новую тарелку на вершину стопки, можете снять верхнюю тарелку со стопки, но только верхнюю. Если вам нужно добраться до нижней тарелки, то есть до самого первого элемента стека, вы должны убрать из стопки все тарелки, причём одну за другой.

То есть если мы положим в стек последовательно числа 11, 22, 33, а потом удалим из стека последний элемент, то в стеке останутся числа 11 и 22. Если мы затем положим в стек числа 44 и 55 и будем последовательно удалять из стека элементы, то элементы будут удаляться в порядке 55, 44, 22, 11.

Стек - структуру данных, которая работает по принципу LIFO (last-in first-out или "последний вошел — первым вышел") — первым всегда извлекается последний добавленный элемент.

Таким образом, стек поддерживает следующие операции:

Добавить элемент в конец стека (push)

Узнать значение последнего элемента стека (top)

Удалить последний элемент из стека (pop)

Узнать размер стека

Все эти операции в стеке выполняются за O(1).

Для реализации стека на C++ подойдёт контейнер vector, так как vector поддерживает операции добавления элемента в конец со средней сложностью O(1) и удаления последнего элемента за O(1) всегда.

Операция push_back(x) эквивалентна операции добавления элемента в конец стека.

Операция pop_back() эквивалентна удалению последнего элемента. Также узнать размер стека нам поможет метод size().

Для того чтобы узнать значение последнего элемента, нужно обратиться к элементу s[s.size() - 1], но у вектора есть метод back(), который возвращает ссылку на последний элемент, удобней использовать его.

Обратите внимание, что вызов методов pop_back() и back() на пустом векторе является ошибкой, в языке C++ это Undefined behaviour, то есть поведение программы после этого не определено. Это может быть как ошибкой исполнения программы, так и продолжением работы программы с произвольными значениями, которые вернули эти методы.

Задача 1. Стек с поддержкой минимального элемента

Давайте реализуем стек, который будет содержать ещё одну операцию — «узнать значение наименьшего элемента во всём стеке». Это так называемый «стек с минимумом». Эта задача решается следующим образом: мы будем хранить два стека: в одном будут сами значения, а в другом стеке для каждого элемента будет храниться минимальное значение во всём стеке, когда мы добавили этот элемент.

Например, пусть мы положили в стек число 55. Оно же будет наименьшим элементом. Теперь положим в стек число 44. Наименьшим элементом во всём стеке теперь стало 44. Положим это число во второй стек. Теперь добавим в стек число 77. Наименьший элемент в стеке до этого — это 44, мы добавили большее число, поэтому значение минимума осталось равно 44, положим его во второй стек. Теперь добавим в стек число 22. Это число меньше минимума во всём стеке, поэтому во второй стек добавим число 22.

Если теперь из первого стека удалять элементы и одновременно удалять элементы из второго стека, то на вершине второго стека всегда будет наименьшее значение в первом стеке.

Реализация функций

```
vector <int> stack, stack_min;
void push(int elem) {
    stack.push_back(elem);
    if (stack_min.empty() || stack_min.back() > elem)
        stack_min.push_back(elem);
    else
        stack_min.push_back(stack_min.back());
}

int top() {
    return stack.back();
}

int get_min() {
    return stack_min.back();
}

void pop() {
    stack.pop_back();
    stack_min.pop_back();
}
```

Задача 2. Задача о правильной скобочной последовательности

Пусть нам дана последовательность из трёх видов скобок: ()(), [][], {}{}. Необходимо проверить, является ли она правильной.

Правильной является последовательность, которая может возникнуть при записи некоторого арифметического выражения.

Пример правильной последовательности: ([]())[]([]())[]

Примеры неправильных последовательностей:

(()(()

(}(}

)()(

([)]([)]

Формально правильная скобочная последовательность определяется таким образом:

- Пустая строка является правильной скобочной последовательностью;

- Правильная скобочная последовательность, взятая в скобки одного типа, - правильная скобочная последовательность;

- Правильная скобочная последовательность, к которой приписана слева или справа правильная скобочная последовательность, - тоже правильная скобочная последовательность.

Можно придумать наивный алгоритм проверки скобочной последовательности на правильность. В правильной скобочной последовательности всегда есть пара скобок, внутри которой нет других скобок. Это пара из открывающей и закрывающей скобки одного вида, идущих рядом. Удалим их. Опять найдём такую пару и снова удалим. Будем продолжать этот процесс, пока есть такие пары. Если мы в итоге смогли удалить все скобки, то есть все скобки разбились на пары открывающих и закрывающих, то последовательность правильная. Если же на каком-то шаге мы не смогли найти пару соседних подходящих скобок, но скобки ещё остались, то последовательность неправильная.

Но такой алгоритм при наивной реализации будет иметь сложность $O(n^2)$. Действительно, давайте рассмотрим последовательность, в которой сначала идут n/2 открывающих скобок одного типа, потом n/2 закрывающих скобок того же типа. Эта последовательность будет правильной. Но каждый раз мы будем удалять две скобки из середины нашей строки, что будет выполняться за O(n), поскольку необходимо сдвигать половину элементов строки. Это удаление элементов из середины будет выполняться n/2 раз, поэтому общая сложность будет O(n2).

Можно улучшить этот алгоритм, если использовать стек. Закрывающая скобка должна быть парной к последней открывающей. То есть если мы будем хранить последовательность из открывающих скобок, то закрывающая скобка удаляет

последнюю открывающую, если она того же вида, в противном случае скобочная последовательность неправильная. Это означает, что последовательность открывающих скобок, которые не были ещё закрыты, представляет собой стек. Если мы встретили открывающую скобку, то она добавляется в конец стека. Если мы встретили закрывающую скобку, то должны выполняться следующие условия: стек не пуст и на вершине стека хранится скобка, парная данной закрывающей скобке. Если после обработки всех скобок стек оказался пуст, то последовательность правильная.

В каком случае последовательность будет неправильной? Если для очередной закрывающей скобки не нашлось парной открывающей, то есть если мы встретили закрывающую скобку, то либо стек пуст, либо на вершине его находится скобка другого вида. Или если мы для какой-то открывающей скобки не нашли закрывающую, это означает, что после рассмотрения всех скобок стек оказался не пуст.

Сложность алгоритма O(n), так как каждую скобку мы кладём в стек или удаляем из стека не более одного раза, а данные операции на стеке работают за O(1).

Дополнительно с понятием стека ознакомьтесь по ссылке: https://metanit.com/cpp/tutorial/7.10.php?ysclid=ls7slezt2q97266816 (C++ | Стек std::stack (metanit.com))

```
#include <iostream>
#include <stack>
#include <string>

using namespace std;

string s;
int n;

int main() {
    int cnt = 0;
    cin >> s;
    n = (int)s.size();
    stack <char> st;
    for (int i = 0; i < n; i++) {
        if (s[i] == '(' || s[i] == '[' || s[i] == '{') {
            st.push(s[i]);
        } else {
            if ((int)st.size() == 0) {
                cout << "no\n";
                return 0;
            }
            if (s[i] == ')' && st.top() == '(') {
                st.pop();
```

```
            } else if (s[i] == '}' && st.top() == '{') {
                st.pop();
            } else if (s[i] == ']' && st.top() == '[') {
                st.pop();
            } else {
                cout << "no\n";
                return 0;
            }
        }
    }
    if ((int)st.size() == 0) {
        cout << "yes\n";
    } else {
        cout << "no\n";
    }
    return 0;
}
```

Задача 3. Стек с защитой от ошибок

Реализуйте структуру данных «стек». Напишите программу, содержащую описание стека и моделирующую работу стека, реализовав все указанные здесь методы. Программа считывает последовательность команд и в зависимости от команды выполняет ту или иную операцию. После выполнения каждой команды программа должна вывести одну строчку. Возможные команды для программы:

- push n - добавить в стек число n (значение n задаётся после команды). Программа должна вывести ok.

- pop - удалить из стека последний элемент. Программа должна вывести его значение.

- back - программа должна вывести значение последнего элемента, не удаляя его из стека.

- size - программа должна вывести количество элементов в стеке.

- clear - программа должна очистить стек и вывести ok.

- exit - программа должна вывести bye и завершить работу.

Перед исполнением операций back и pop программа должна проверять, содержится ли в стеке хотя бы один элемент. Если во входных данных встречается операция back или pop и при этом стек пуст, то программа должна вместо числового значения вывести строку error.

Реализация

```
#include <iostream>
#include <vector>
#include <string>

using namespace std;

string s;
int n, x;

vector <int> ve;

void push(int x) {
    ve.push_back(x);
    cout << "ok\n";
}

void pop() {
    if ((int)ve.size() == 0) {
        cout << "error\n";
    }
    else {
        cout << ve.back() << '\n';
        ve.pop_back();
    }
}

void back() {
    if ((int)ve.size() == 0) {
        cout << "error\n";
    }
    else {
        cout << ve.back() << '\n';
    }
}

void size() {
    cout << (int)ve.size() << '\n';
}

void clear() {
    ve.clear();
    cout << "ok\n";
}

int main() {
    while (cin >> s) {
        if (s == "exit") {
            cout << "bye\n";
            return 0;
        }
        if (s == "push") {
            cin >> x;
            push(x);
        }
        if (s == "pop") {
            pop();
        }
        if (s == "back") {
            back();
        }
        if (s == "size") {
            size();
        }
        if (s == "clear") {
            clear();
```

```
        }
    }
    return 0;
}
```

2. Очередь

Другая структура данных, похожая на стек, — это очередь (queue). Её назначение понятно из названия. Элементы выстраиваются один за другим, новые элементы добавляются в конец очереди, а удаляются элементы из начала очереди. Данная структура работает по принципу «первый пришёл — первый ушёл» — FIFO.

Очередь поддерживает следующие операции:

- Добавить элемент в конец очереди (push)
- Удалить элемент из начала очереди (pop)
- Узнать значение первого элемента очереди (front)
- Узнать размер очереди (size)

Все операции с очередью должны выполняться за O(1).

Подумаем, как можно хранить очередь. Давайте, как в случае со стеком, будем хранить элементы очереди в массиве, добавляя новые элементы в конец массива. Но чтобы не перемещать все элементы очереди при удалении первого элемента, будем просто считать, что начало очереди смещается, то есть позиция удаляемого элемента не используется. Мы будем запоминать позицию, где начинается очередь. Тогда при операциях удаления элементов из очереди очередь будет двигаться дальше от начала массива, а место удалённых элементов будет потерянной областью памяти. Чтобы бороться с этой проблемой, есть два подхода. Один подход: давайте логически замкнём массив, в котором будет храниться очередь в кольцо, то есть за последним элементом будет идти первый. Тогда когда очередь доползёт до конца выделенного массива, она начнёт расти с начала, то есть мы будем повторно использовать места, занятые удалёнными элементами. Размер такого массива должен позволять разместить столько элементов, сколько одновременно могут храниться в очереди в любой момент времени, но не обязательно - все добавленные в очередь элементы.

Другой подход - регулярное сжатие свободного места. Если в начале очереди накопилось слишком много неиспользованных ячеек (например, столько же, сколько элементов в самой очереди), то можно сразу удалить все эти ячейки, скопировав все

элементы очереди в начало массива. То есть мы будем операцию реального удаления элементов выполнять редко, но удалять сразу же много элементов, тогда средняя сложность в пересчёте на одну операцию удаления будет O(1).

Есть и другой интересный способ реализации очереди. Если последовательность элементов положить в стек, а потом излечь, то она развернётся в обратном порядке. Если это повторить ещё раз, то последовательность элементов опять развернётся и порядок будет исходным. Очередь же не меняет порядок элементов, поэтому можно реализовать очередь с использованием двух стеков.

Возьмём два стека: s1 и s2. Операцию добавления будем всегда делать в первый стек. А удалять элементы будем из второго стека. Но при этом второй стек изначально пустой, все добавленные элементы будут находиться в первом стеке. Если необходимо удалить элемент, а второй стек пуст, то все элементы из первого стека перенесём во второй стек. При этом самый первый добавленный элемент, который был внизу первого стека, окажется на вершине второго стека и выйдет из очереди первым. Если s2 не пуст, достаём элементы из него. Как только s2 окажется снова пустым, повторяем ту же операцию.

Кажется, что при таком подходе изменится вычислительная сложность операций очереди, потому что операция удаления элементов может выполняться долго, так как придётся все элементы перекладывать из одного стека в другой. Но такие операции будут происходить редко, и средняя сложность одной операции оказывается невелика. Действительно, пусть мы n элементов добавили в очередь и n элементов удалили, как-то чередуя эти операции. Тогда каждый элемент сначала был добавлен в стек s1, потом перенесён в стек s2, потом удалён из стека s2, то есть мы выполнили 2 операции добавления элемента в стек и 2 операции удаления, всего - 4n операций со стеком для реализации 2n операций с очередью. Итого на одну операцию с очередью в среднем требуется O(1) операций со стеком.

Эта идея может показаться сложной для применения на практике, но мы используем её для реалиазации другой структуры - очереди с минимумом или максимумом, то есть очереди, которая поддерживает ещё один запрос - узнать значение наименьшего или наибольшего элемента в очереди. Сложно реализовать операцию взятия минимума, когда из структуры данных удаляется один элемент. Пусть мы знали,

что наименьшее значение всех элементов в структуре равно m и из структуры был удалён элемент, равный m. Какое стало теперь наименьшее значение во всей структуре? Оно может оказаться как m, так и больше m.

Но раньше мы рассмотрели, как реализовать стек с минимумом, теперь для реализации очереди с минимумом мы будет использовать два стека с минимумом, то есть минимумы в стеках будут поддерживаться автоматически. Чтобы узнать значение минимума в очереди, нужно взять минимум из минимальных значений в каждом из двух стеков.

Задача 4. Очередь с защитой от ошибок

Реализуйте структуру данных «очередь». Напишите программу, содержащую описание очереди и моделирующую работу очереди, реализовав все указанные здесь методы. Программа считывает последовательность команд и в зависимости от команды выполняет ту или иную операцию. После выполнения каждой команды программа должна вывести одну строчку. Возможные команды для программы:

- push n - добавить в очередь число n (значение n задаётся после команды). Программа должна вывести ok;

- pop - удалить из очереди первый элемент. Программа должна вывести его значение;

- front - программа должна вывести значение первого элемента, не удаляя его из очереди;

- size - программа должна вывести количество элементов в очереди;

- clear - программа должна очистить очередь и вывести ok;

- exit - программа должна вывести bye и завершить работу.

Перед исполнением операций front и pop программа должна проверять, содержится ли в очереди хотя бы один элемент. Если во входных данных встречается операция front или pop и при этом очередь пуста, то программа должна вместо числового значения вывести строку error.

Реализация

```
#include <iostream>
#include <string>
#include <deque>

using namespace std;
```

```
string s;
int n, x;

deque <int> ve;

void push(int x) {
    ve.push_back(x);
    cout << "ok\n";
}

void pop() {
    if ((int)ve.size() == 0) {
        cout << "error\n";
    }
    else {
        cout << ve.front() << '\n';
        ve.pop_front();
    }
}

void front() {
    if ((int)ve.size() == 0) {
        cout << "error\n";
    }
    else {
        cout << ve.front() << '\n';
    }
}

void size() {
    cout << (int)ve.size() << '\n';
}

void clear() {
    ve.clear();
    cout << "ok\n";
}

int main() {
    while (cin >> s) {
        if (s == "exit") {
            cout << "bye\n";
            return 0;
        }
        if (s == "push") {
            cin >> x;
            push(x);
        }
        if (s == "pop") {
            pop();
        }

        if (s == "size") {
            size();
        }
        if (s == "front") {
            front();
        }
        if (s == "clear") {
            clear();
        }
    }
    return 0;

}
```

Дополнительно ознакомиться с очередью можно по ссылке

https://metanit.com/cpp/tutorial/7.11.php?ysclid=ls7uwdpjrg974209464

(C++ | Очередь std::queue (metanit.com))

3. Дек

Если есть стек, в который элементы кладутся в один конец и извлекаются с того же конца, и очередь, в которой элементы кладутся в один конец, а извлекаются с другого конца, то логичным кажется существование структуры данных, которая называется дек, по английски deque или double-ended queue.

Дек поддерживает следующие операции:

- Добавить элемент в начало дека
- Добавить элемент в конец дека
- Удалить элемент из начала дека
- Удалить элемент из конца дека
- Узнать значение последнего элемента дека
- Узнать значение первого элемента дека
- Узнать размер дека

Операции получения значения элемента и удаления элемента могут быть совмещены. Все операции должны выполняться за O(1) или в среднем за O(1).

Дек можно реализовать так же, как и очередь, - на закольцованном массиве, но необходимо учесть, что при добавлении элементов дек может ползти в любом из двух направлений, поэтому реализация более сложна, чем реализация очереди.

Для реализации на C++ существует специальный контейнер deque. Он находится в библиотеке deque.

Приведём пример работы с деком:

```
#include <deque>
#include <iostream>
using namespace std;
int main()
{
    deque<int> D;
    D.push_back(5);
    D.push_front(4);
    cout << D.size() << " " << D.front() << " " << D.back() << endl; //
2 4 5
    D.push_front(6);
    D.pop_back();
```

```
    cout << D.size() << " " << D.front() << " " << D.back() << endl; //
2 6 4
    return 0;
}
```

У дека в C++ есть следующие операции:

push_back() — добавить элемент в конец дека

push_front() — добавить элемент в начало дека

pop_back() — удалить элемент из конца дека

pop_front() — удалить элемент из начала дека

back() — узнать значение последнего элемента дека

front() — узнать значение первого элемента дека

size() — узнать размер дека

empty() — проверка дека на пустоту

clear() — очистить дек

Интересной особенностью реализации дека в C++ является то, что вы можете обратиться к элементу дека по индексу, как к элементам массива. Например, если в деке d будет 55 элементов, то к ним можно обращаться как d[0],d[1],....,d[4]. Это обращение будет выполняться за O(1), но будет чуть медленнее, чем обращение к элементам вектора или массива. На практике этот же контейнер можно рекомендовать и для работы с очередью — собственный контейнер для работы с очередью реализован именно через дек.

Задача 5. Дек с защитой от ошибок

Реализуйте структуру данных «дек». Напишите программу, содержащую описание дека и моделирующую работу дека, реализовав все указанные здесь методы. Программа считывает последовательность команд и в зависимости от команды выполняет ту или иную операцию. После выполнения каждой команды программа должна вывести одну строчку. Возможные команды для программы:

- push_front - добавить (положить) в начало дека новый элемент. Программа должна вывести ok;

- push_back - добавить (положить) в конец дека новый элемент. Программа должна вывести ok;

- pop_front - извлечь из дека первый элемент. Программа должна вывести его значение;

- pop_back - извлечь из дека последний элемент. Программа должна вывести его значение;

- front - узнать значение первого элемента (не удаляя его). Программа должна вывести его значение;

- back - узнать значение последнего элемента (не удаляя его). Программа должна вывести его значение;

- size - вывести количество элементов в деке;

- clear - очистить дек (удалить из него все элементы) и вывести ok;

- exit - программа должна вывести bye и завершить работу.

Гарантируется, что количество элементов в деке в любой момент не превосходит 100. Перед исполнением операций pop_front, pop_back, front, back программа должна проверять, содержится ли в деке хотя бы один элемент. Если во входных данных встречается операция pop_front, pop_back, front, back и при этом дек пуст, то программа должна вместо числового значения вывести строку error.

Реализация

```
#include <iostream>
#include <string>
#include <deque>

using namespace std;

string s;
int n, x;

deque <int> ve;

void push_back(int x) {
   ve.push_back(x);
   cout << "ok\n";
}

void push_front(int x) {
   ve.push_front(x);
   cout << "ok\n";

}

void pop_front(int x) {
   if ((int)ve.size() == 0) {
      cout << "error\n";
   }
   else {
      cout << ve.front() << '\n';
      ve.pop_front();
   }
```

```
}

void pop_back() {
    if ((int)ve.size() == 0) {
        cout << "error\n";
    }
    else {
        cout << ve.back() << '\n';
        ve.pop_back();
    }
}

void front() {
    if ((int)ve.size() == 0) {
        cout << "error\n";
    }
    else {
        cout << ve.front() << '\n';
    }
}

void back() {
    if ((int)ve.size() == 0) {
        cout << "error\n";
    }
    else {
        cout << ve.back() << '\n';
    }
}

void size() {
    cout << (int)ve.size() << '\n';
}

void clear() {
    ve.clear();
    cout << "ok\n";
}

int main() {
    while (cin >> s) {
        if (s == "exit") {
            cout << "bye\n";
            return 0;
        }
        if (s == "push_front")
        {
            cin >> x;
            push_front(x);
        }
        if (s == "push_back")
        {
            cin >> x;
            push_back(x);
        }
        if (s == "pop_front") {
            pop_front(x);
        }

        if (s == "pop_back") {
            pop_back();
        }

        if (s == "size") {
            size();
        }
        if (s == "front") {
```

```
            front();

        }

        if (s == "back") {

            back();
        }
        if (s == "clear") {
            clear();
        }
    }
    return 0;
}

}
```

4. Динамическая структура. Список

Под динамической структурой данных понимается любая структура данных, занимаемый объем памяти которой не является фиксированным. Иными словами, в подобной структуре может храниться как два, пять, двадцать элементов, так и одно большое ничего. Размер подобной структуры ограничен только объемом оперативной памяти компьютера.

Существует несколько разновидностей динамических структур: список, дерево.

Прежде чем переходить к описанию структур, следует запомнить несколько простых определений:

Потомок - элемент структуры, идущий после текущего. В зависимости от вида динамической структуры у элемента может быть более одного потомка.

Предок - элемент структуры, идущий до текущего.

Головной элемент (Head) - первый элемент списка.

Хвостовой элемент (Tail) - последний элемент списка.

Список — это линейная динамическая структура данных, у каждого элемента может быть только один предок и только один потомок. По сути своей это очень похоже на обыкновенный массив, с той лишь разницей, что размер его не имеет ограничений. Списки также подразделяются на несколько типов.

Односвязный список — элемент имеет указатель только на своего потомка.

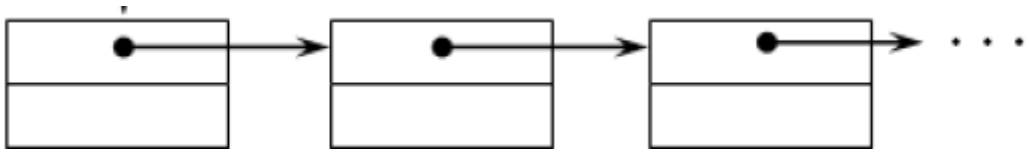

Рисунок 1 - Односвязный список

Двусвязный список - элемент имеет указатели и на потомка, и на родителя.

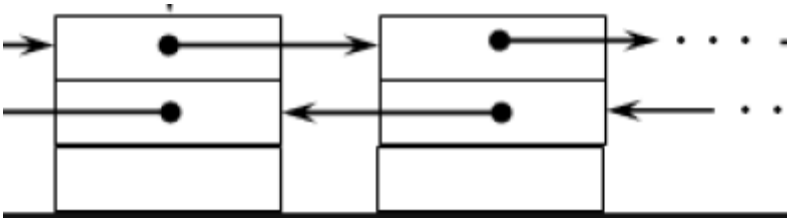

Рисунок 2 -Двусвязный список

Замкнутый (кольцевой, циклический) список - головной и хвостовой элементы которого указывают друг на друга.

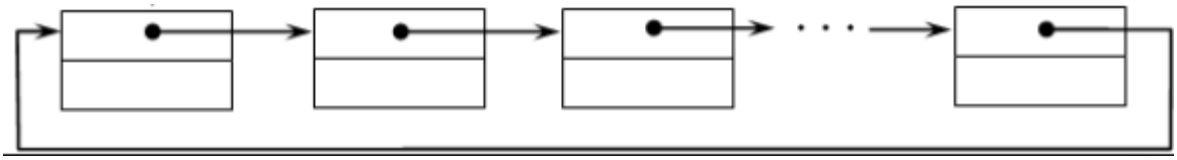

Рисунок 3 - Замкнутый список

На базе простого однонаправленного списка могут быть построены такие структуры данных, как очередь (queue) и стек (stack).

Дополнительная информация о списках по ссылке https://metanit.com/cpp/tutorial/7.6.php (C++ | List (metanit.com))

Практическая часть

Базовый уровень

Задание 1: Написать программу в соответствии с индивидуальным заданием

№ вар.	Задача
1	1. Создать стек из целых чисел. Вычислить произведение нечётных значений элементов стека. Организовать просмотр данных стека. 2. Создать стек, информационными полями которого являются: монитор, диагональ и его цена. Добавить в стек сведения о новом мониторе. Организовать просмотр данных стека и определить количество мониторов с диагональю более 20 дюймов.
2	1. Создать очередь из вещественных чисел. Определить количество положительных значений элементов очереди. Организовать просмотр данныхочереди. 2. Создать стек, информационными полями которого являются: наименование товара и его цена. Добавить в стек сведения о новом товаре. Организовать просмотр данных стека и вычислить среднюю цену товаров.
3	1. Дано число N (> 0) и набор из N чисел. Создать стек, содержащий исходные числа (последнее число будет вершиной стека), и вывести указательна его вершину. 2. Создать очередь из вещественных чисел. Определить количество отрицательных чисел очереди. Организовать просмотр данных очереди
4	1. Создать очередь из вещественных чисел. Определить минимальный элемент очереди. Организовать просмотр данных очереди. 2. Создать очередь, информационными полями которого являются: компьютер и объем его оперативной памяти. Удалить из очереди сведения окомпьютере, которые были введены первыми. Организовать просмотрданных очереди и вычислить общий объем памяти компьютеров, записанных в очереди.
5	1. Дано число D и указатели P_1 и P_2 на начало и конец очереди, содержащей не менее двух элементов. Добавить элемент со значением D в конецочереди и извлечь из очереди первый (начальный) элемент. Вывести значение извлеченного элемента и содержимое очереди. После извлечения элемента из очереди освободить память, занимаемую этим элементом. 2. Создать очередь, информационными полями которой являются: наименование процессора и его тактовая частота и количество ядер. Добавить в очередь сведения о новом процессоре. Организовать просмотр данных очереди и распечатать данные о многоядерных процессорах (количество ядер больше 1).
6	1. Создать стек из вещественных чисел. Определить максимальный элемент в стеке. Организовать просмотр данных стека. 2. Создать стек, информационными полями которого являются: книга и её цена. Добавить в стек сведения о новой книге. Организовать просмотр данных стека и вычислить среднюю цену книг.

7	1. Создать стек, информационными полями которого являются: фамилия исредний бал студента. Добавить в стек сведения о новом студенте. Организовать просмотр данных стека. 2. Создать очередь, информационными полями которой являются: наименование товара и его стоимость. Добавить в очередь сведения о новом товаре. Организовать просмотр данных очереди и вычислить общуюстоимость товаров с наименованием «Ручка шариковая».
8	1. Создать стек из целых чисел. Вычислить среднее арифметическое чётных значений элементов стека. Организовать просмотр данных стека. 2. Создать очередь, информационными полями которой являются: телефони его цена. Удалить из очереди сведения о телефоне, которые были введены первыми. Организовать просмотр данных очереди.
9	1. Создать очередь из целых чисел. Определить количество положительныхэлементов очереди. Организовать просмотр данных очереди. 2. Создать стек, информационными полями которого являются: название горы и высота. Добавить в стек сведения о новой горе. Организовать просмотр данных стека и определить среднюю высоту гор.
10	1. Создать очередь из целых чисел. Определить среднее значение элементов очереди. Организовать просмотр данных очереди. 2. Создать стек, информационными полями которого являются: название книги и количество страниц. Добавить в стек сведения о новой книге. Организовать просмотр данных стека и определить количество книг в стеке.
11	1. Создать очередь из целых чисел. Определить количество четных значений элементов очереди. Организовать просмотр данных очереди. 2. Создать очередь, информационными полями которой являются: фамилия и средний бал студента. Добавить в очередь сведения о новом студенте. Организовать просмотр данных очереди.
12	1. Создать стек из вещественных чисел. Определить максимальный элемент в стеке. Организовать просмотр данных стека. 2. Создать стек, информационными полями которого являются фамилия работника и его оклад. Добавить в стек сведения о новом работника. Организовать просмотр данных стека и вычислить средний оклад.
13	1. Дано число N (> 0) и указатели P_1 и P_2 на начало и конец непустой очереди. Извлечь из очереди N начальных элементов и вывести их значения(если очередь содержит менее N элементов, то извлечь все ее элементы). После извлечения элементов из очереди освобождать память, которую они занимали. 2. Создать очередь, информационными полями которой являются: длины катетов прямоугольного треугольника (два вещественных числа). Добавить в очередь сведения о новом треугольнике. Организовать просмотр данных очереди. Определить периметр треугольника в начале

	очереди
14	1. Создать очередь из целых чисел. Определить количество элементов очереди меньших 10. Организовать просмотр данных очереди. 2. Создать очередь, информационными полями которой являются: наименование процессора и его тактовая частота и количество ядер. Добавить в очередь сведения о новом процессоре. Организовать просмотр данных очереди и распечатать данные о многоядерных процессорах (количество ядер больше 1).
15	1. Создать очередь из целых чисел. Определить количество простых чисел. Организовать просмотр данных очереди. 2. Создать стек, информационными полями которого являются: улица, номер дома и номер квартиры. Добавить в стек сведения о новой квартире. Организовать просмотр данных стека и определить количество домов на улице «Дерибасовская».

Средний уровень

Задание 2: написать программу в соответствии с индивидуальным заданием

№ вар.	Задача
1	1. Создать стек целочисленных значений, для реализации используя односвязные списки. Реализовать операции добавления (push) и удаления (pop) элемента из стека. Добавьте в стек числа 4, 3, 1, 2, 4 и распечатайте содержимое стека. Удалите один элемент из стека, и распечатайте содержимое стека еще раз. Найдите минимальный элемент, принадлежащий стеку. 2. Даны две непустые очереди; адреса начала и конца первой равны P_1 и P_2, а второй — P_3 и P_4. Очереди содержат одинаковое количество элементов. Объединить очереди в одну, в которой элементы исходных очередей чередуются (начиная с первого элемента первой очереди). Вывести указатели на начало и конец полученной очереди. Операции выделения и освобождения памяти не использовать
2	1. Создать очередь вещественных значений, для реализации используя односвязные списки. Реализовать операции добавления (enqueue) и удаления (dequeue) элемента из очереди. Добавьте в очередь числа: -2.2, 2.3, 2.2, 5.1, 6.7 и распечатайте содержимое очереди. Удалите 3 элемента из очереди, затем добавьте в очередь число 1.9 и распечатайте очередь еще раз. Найдите произведение элементов, принадлежащих очереди.

	2. Создать стек целочисленных значений, для реализации используя односвязные списки. Реализовать операции добавления (push) и удаления (pop) элемента из стека. Добавьте в стек числа 1, 2, -3, 4 и распечатайте содержимое стека. Удалите один элемент из стека, и распечатайте содержимое стека еще раз. * Найдите максимальный элемент, принадлежащий стеку.
3	1. Создать стек строковых значений, для реализации используя односвязные списки. Реализовать операции добавления (push) и удаления (pop) элемента из стека. Добавьте в стек строки «abc», «fx», «glc», «hi», «gogo» и распечатайте содержимое стека. Удалите один элемент из стека, затем добавьте строку «the end» и распечатайте содержимое стекаеще раз. Найдите количество строк в стеке, состоящих из 2 символов 2. Создать очередь строковых значений, для реализации используя односвязные списки. Реализовать операции добавления (enqueue) и удаления(dequeue) элемента из очереди. Добавьте в очередь строки «one», «two», «three», «four», «five», «six», и распечатайте содержимое очереди. Удалите 2 элемента из очереди, затем добавьте в очередь строку «seven»и распечатайте очередь еще раз. Найдите количество строк начинающихся с букв «f» или «t».
4	1. Создать очередь строковых значений, для реализации используя односвязные списки. Реализовать операции добавления (enqueue) и удаления(dequeue) элемента из очереди. Добавьте в очередь строки «one», «two», «three», «four» и распечатайте содержимое очереди. Удалите 2 элемента из очереди, затем добавьте в очередь строку «inf» и распечатайте очередь еще раз. Найдите суммарную длину строк, принадлежащих очереди, кроме первой строки очереди. 2. Создать стек целочисленных значений, для реализации используя односвязные списки. Реализовать операции добавления (push) и удаления (pop) элемента из стека. Добавьте в стек числа -5, 3, -4, 5 и распечатайте содержимое стека. Удалите один элемент из стека, добавьте в стек число 10, и распечатайте стек еще раз. * Найдите сумму всех положительных элементов, принадлежащих стеку.
5	1. Создать стек целочисленных значений, для реализации используя односвязные списки. Реализовать операции добавления (push) и удаления (pop) элемента из стека. Добавьте в стек числа 1, 2, 3, 4, 5 и распечатайте содержимое стека. Удалите 3 элемента из стека, и распечатайте содержимое стека еще раз. Найдите сумму элементов стека. 2. Создать очередь вещественных значений, для реализации используя односвязные списки. Реализовать операции добавления (enqueue) и удаления (dequeue) элемента из очереди. Добавьте в очередь числа 46.5, 3.4, 32.4, -3.21 и распечатайте содержимое очереди. Удалите 2 элемента из очереди, затем добавьте в очередь число 5.0 и распечатайте очередь еще раз. Найдите сумму элементов, по модулю больших 12, принадлежащих очереди.

6	1. Создать очередь вещественных значений, для реализации используя односвязные списки. Реализовать операции добавления (enqueue) и удаления (dequeue) элемента из очереди. Добавьте в очередь числа 2.1, 2.1, 5.3 и распечатайте содержимое очереди. Удалите 1 элемент изочереди, затем добавьте в очередь число 4.9 и распечатайте очередь ещераз. Найдите сумму элементов очереди 2. Создать стек строковых значений, для реализации используя односвязные списки. Реализовать операции добавления (push) и удаления (pop) элемента из стека. Добавьте в стек строки «Students», «of», «the», «group», «ТЕ», «3» и распечатайте содержимое стека. Удалите одинэлемент из стека, и распечатайте содержимое стека еще раз. Напечатайте все строки, которые состоят из двух символов из стека.
7	1. Создать очередь вещественных значений, для реализации используя односвязные списки. Реализовать операции добавления (enqueue) и удаления (dequeue) элемента из очереди. Добавьте в очередь числа 2.2, 1.2, 2.0, 5.2 и распечатайте содержимое очереди. Удалите 2 элемента из очереди, затем добавьте в очередь число 2.9 и распечатайте очередь еще раз. Найдите сумму элементов очереди. 2. Создать очередь строковых значений, для реализации используя односвязные списки. Реализовать операции добавления (enqueue) и удаления(dequeue) элемента из очереди. Добавьте в очередь строки «one», «two», «three», «four», «five», «six», «seven» и распечатайте содержимое очереди. Удалите 4 элемента из очереди, затем добавьте в очередь строки «eight», «nine» и распечатайте очередь еще раз. Найдите количество строк, состоящих из 4 символов.
8	1. Создать стек строковых значений, для реализации используя односвязные списки. Реализовать операции добавления (push) и удаления (pop) элемента из стека. Добавьте в стек строки «sdf», «2», «ssd4», «hello» и распечатайте содержимое стека. Удалите 2 элемента из стека, и распечатайте содержимое стека еще раз. Найдите строку минимальной длины, принадлежащую стеку. 2. Создать стек целочисленных значений, для реализации используя односвязные списки. Реализовать операции добавления (push) и удаления (pop) элемента из стека. Добавьте в стек числа 1, 4, 2 и распечатайте содержимое стека. Добавьте число 4 в стек, и распечатайте содержимое стека еще раз. * Найдите количество чисел, больших числа 3, принадлежащий стеку.
9	1. Создать очередь строковых значений, для реализации используя односвязные списки. Реализовать операции добавления (enqueue) и удаления(dequeue) элемента из очереди. Добавьте в очередь строки «one», «two», «three», «four» и распечатайте содержимое очереди. Удалите 1 элемент из очереди, затем добавьте в очередь строку «five» и распечатайте очередь еще раз. Найдите суммарную длину всех

	строк, принадлежащих очереди. 2. Создать очередь вещественных значений, для реализации используя односвязные списки. Реализовать операции добавления (enqueue) и удаления (dequeue) элемента из очереди. Добавьте в очередь числа –2.1, 1.3, -1.34, 3.3 и распечатайте содержимое очереди. Удалите 1 элемент изочереди, затем добавьте в очередь число 2.9 и распечатайте очередь еще раз. Найдите сумму отрицательных элементов очереди.
10	1. Создать очередь вещественных значений, для реализации используя односвязные списки. Реализовать операции добавления (enqueue) и удаления (dequeue) элемента из очереди. Добавьте в очередь числа 2.2, 3.2, 2.4, -3.2 и распечатайте содержимое очереди. Удалите 1 элемент из очереди, затем добавьте в очередь число 0.04 и распечатайте очередь еще раз. * Найдите сумму чисел по модулю меньших 1, принадлежащих очереди. 2. Создать стек строковых значений, для реализации используя односвязные списки. Реализовать операции добавления (push) и удаления (pop) элемента из стека. Добавьте в стек строки «lll», «2», «sdf4», «bye» и распечатайте содержимое стека. Удалите 1 элемент из стека, и распечатайте содержимое стека еще раз. Найдите строку максимальной длины, принадлежащую стеку.
11	1. Создать стек целочисленных значений, для реализации используя односвязные списки. Реализовать операции добавления (push) и удаления (pop) элемента из стека. Добавьте в стек числа -5, 3, -4, 5 и распечатайте содержимое стека. Удалите один элемент из стека, добавьте число 10 в стек, и напечатайте содержимое стека еще раз. Найдите суммувсех положительных элементов, принадлежащих стеку 2. Создать очередь строковых значений, для реализации используя односвязные списки. Реализовать операции добавления (enqueue) и удаления (dequeue) элемента из очереди. Добавьте в очередь строки «one», «two», «three», «four» и распечатайте содержимое очереди. Добавьте в очередь строку «five» и распечатайте очередь еще раз. * Найдите суммарную длину всех строк, принадлежащих очереди, кроме последней строкиочереди.
12	1. Создать стек строковых значений, для реализации используя односвязные списки. Реализовать операции добавления (push) и удаления (pop) элемента из стека. Добавьте в стек строки «Students», «of», «the», «group», «ТЕ» и распечатайте содержимое стека. Удалите один элемент из стека, и распечатайте содержимое стека еще раз. Напечатайте все строки, начинающиеся со строчной буквы «t», принадлежащие стеку. 2. Создать стек целочисленных значений, для реализации используя односвязные списки. Реализовать операции добавления (push) и удаления (pop) элемента из стека. Добавьте в стек числа 5, 3, 44, 555 и распечатайте содержимое стека. Удалите 2 элемента из стека, добавьте числа 20 и 30 в стек, и напечатайте содержимое стека

	еще раз. Найдите количество положительных двухзначных чисел, принадлежащих стеку.
13	1. Создать очередь строковых значений, для реализации используя односвязные списки. Реализовать операции добавления (enqueue) и удаления(dequeue) элемента из очереди. Добавьте в очередь строки «one», «two», «three», «four», «five», «six», «seven» и распечатайте содержимое очереди. Удалите 1 элемент из очереди, затем добавьте в очередь строку «eight» и распечатайте очередь еще раз. Найдите количество строкначинающихся с букв «*s*» или «*t*». 2. Создать очередь вещественных значений, для реализации используя односвязные списки. Реализовать операции добавления (enqueue) и удаления (dequeue) элемента из очереди. Добавьте в очередь числа -2.2, 5.5, 4.3, -4.5 и распечатайте содержимое очереди. Удалите 1 элемент из очереди и распечатайте очередь еще раз. Найдите сумму чисел по модулю больше 4, принадлежащих очереди.
14	1. Создать стек строковых значений, для реализации используя односвязные списки. Реализовать операции добавления (push) и удаления (pop) элемента из стека. Добавьте в стек строки «abc», «de», «f», «g», «hi», «jk» и распечатайте содержимое стека. Удалите один элемент из стека, ираспечатайте содержимое стека еще раз. * Найдите количество односимвольных строк в стеке. 2. Создать стек строковых значений, для реализации используя односвязные списки. Реализовать операции добавления (push) и удаления (pop) элемента из стека. Добавьте в стек строки «Student», «of», «the», «OSAT» и распечатайте содержимое стека. Удалите один элемент из стека, добавьте строку «ONAT» и распечатайте содержимое стека еще раз. * Посчитайте количество строк, состоящих не менее чем из трех символов, принадлежащих стеку.
15	1. Создать очередь строковых значений, для реализации используя односвязные списки. Реализовать операции добавления (enqueue) и удаления(dequeue) элемента из очереди. Добавьте в очередь строки «one», «two», «three», «four» и распечатайте содержимое очереди. Удалите 1 элемент из очереди, затем добавьте в очередь строку «five» и распечатайте очередь еще раз. Найдите суммарную длину строк, принадлежащих очереди. 2. Создать очередь вещественных значений, для реализации используя односвязные списки. Реализовать операции добавления (enqueue) и удаления (dequeue) элемента из очереди. Добавьте в очередь числа -2.0, 2.0, 2.1, -2.1, 3.7 и распечатайте содержимое очереди. Удалите 2 элемента из очереди, затем добавьте в очередь число 1.1 и распечатайте очередь еще раз. Найдите произведение

	положительных элементов, принадлежащих очереди.

Высокий уровень

Задание 3: Написать программу в соответствии с индивидуальным заданием

№ вар.	Задача
1	Даны две непустые очереди; адреса начала и конца первой равны P_1 и P_2, а второй — P_3 и P_4. Элементы каждой из очередей упорядочены по возрастанию (в направлении от начала очереди к концу). Объединить очереди в одну с сохранением упорядоченности элементов. Вывести указатели на начало и конец полученной очереди. Операции выделения и освобождения памяти не использовать, поля с данными (Data)не изменять.
2	Арифметическое выражение можно представить в обратной польской записи, где знаки операции следуют за операндами (а не ставятся между ними, как в обычной записи выражений). Обратная польская запись не требует скобок. Например, выражению «1 + 2» соответствует запись «1 2 +», выражению «1 + 2 * 3» запись «1 2 3 * +» (вначале умножаются 2 на 3, а потом 1 складывается с результатом), «(2 + 3) * (3 – 1)» записываетсякак «2 3 + 3 1 – *». Задается строка – выражение в обратной польскойзаписи (числа и знаки +, –, * разделены пробелами). Используя стек, вычислите значение выражения. **Подсказка**: нужно последовательно перебрать все числа и знаки из строки, числа нужно заносить в стек, а как встретится знак операции, вынимать 2 числа из стека, применять к ним текущую операцию, а результат заносить в стек.
3	Дана последовательность скобок вида «(», «)», «[», «]», «{», «}». Правильными скобочными последовательностями называются пустая последовательность, а также «(P)», «[P]», «{P}», где P – некоторая правильная последовательность. Например «{}()[]» и «{[][()()]}()» – правильные скобочные последовательности, а «(]», «[()» и «({)}» – неправильные. Определите является ли заданная строка правильным скобочным выражением. **Подсказка**: обработайте по очереди все символы входной строки, помещая открывающие скобки в стек, а для закрывающих скобок извлекайте открывающую скобку из стека и проверяйте, соответствуют ли они друг другу.

4	Реализуйте очередь, используя два стека. **Подсказка**: стек инвертирует порядок элементов (т.е. если добавить 1 2 3 4 5, вынимая элементы, получим 5 4 3 2 1), поэтому если элементы пройдут два стека, то восстановится их исходный порядок. При помещении в очередь помещайте элемент в первый стек, при извлечении элемента из очереди, извлекайте элемент из второго стека, а если при этом второй стек окажется пустым, перенесите *все* элементы из первого стека во второй (доставайте элементы первого стека, пока они есть, и переносите во второй).
5	Дано число N (> 0) и две непустые очереди; адреса начала и конца первой равны P_1 и P_2, а второй — P_3 и P_4. Переместить N начальных элементов первой очереди в конец второй очереди. Если первая очередь содержит менее N элементов, то переместить из первой очереди во вторую все элементы. Вывести новые адреса начала и конца первой, а затем второй очереди (для пустой очереди дважды вывести nil). Операции выделения и освобождения памяти не использовать.
6	Дан набор из 10 чисел. Создать две очереди: первая должна содержать числа из исходного набора с нечетными номерами (1, 3, …, 9), а вторая — с четными (2, 4, …, 10); порядок чисел в каждой очереди должен совпадать с порядком чисел в исходном наборе. Вывести указатели на начало и конец первой, а затем второй очереди.
7	Дан набор из 10 чисел. Создать две очереди: первая должна содержать все нечетные, а вторая — все четные числа из исходного набора (порядок чисел в каждой очереди должен совпадать с порядком чисел в исходном наборе). Вывести указатели на начало и конец первой, а затем второй очереди (одна из очередей может оказаться пустой; в этом случае вывестидля нее две константы nil).
8	Даны указатели P_1 и P_2 на начало и конец непустой очереди. Извлекать из очереди элементы, пока значение начального элемента очереди не станет четным, и выводить значения извлеченных элементов (если очередь не содержит элементов с четными значениями, то извлечь все ее элементы). Вывести также новые адреса начала и конца очереди (для пустой очереди дважды вывести nil). После извлечения элементов из очереди освобождать память, которую они занимали.
9	Даны две очереди; адреса начала и конца первой равны P_1 и P_2, а второй – P_3 и P_4 (если очередь является пустой, то соответствующие адреса равны nil). Переместить все элементы первой очереди (в порядке от началак концу) в конец второй очереди и вывести новые адреса начала и конца второй очереди. Операции выделения и освобождения памяти не использовать.

10	Даны две непустые очереди; адреса начала и конца первой равны P_1 и P_2, а второй – P_3 и P_4. Перемещать элементы из начала первой очередив конец второй, пока значение начального элемента первой очереди не станет четным (если первая очередь не содержит четных элементов, то переместить из первой очереди во вторую все элементы). Вывести новые адреса начала и конца первой, а затем второй очереди (для пустой очереди дважды вывести nil). Операции выделения и освобождения памяти не использовать.

Динамические структуры: списки

Базовый уровень

Задание 4: написать программу в соответствии с индивидуальным заданием

№ вар.	Задача
1	1. Создать линейный однонаправленный список из целых чисел. Определить среднее арифметическое значений элементов списка, кратных 4. 2. Создать линейный однонаправленный список строк. Удалить строку с минимальной длиной.
2	1. Создать линейный однонаправленный список из целых чисел. Определить сумму значений элементов списка, кратных 5. 2. Создать линейный однонаправленный список строк. Вывести все строки списка, начинающиеся на заглавную латинскую букву.
3	1. Создать линейный однонаправленный список из вещественных чисел. Определить количество элементов списка со значениями больше 7. 2. Создать линейный однонаправленный список из целых чисел. Вставить в список число 15 после каждого элемента, по модулю меньшего 2.
4	1. Создать линейный однонаправленный список из целых чисел. Вставить в список число 10 после первого элемента с отрицательным значением. 2. Создать линейный однонаправленный список из целых чисел. Удалить из списка два элемента после каждого элемента с нечетным значением.
5	1. Создать линейный однонаправленный список из вещественных чисел. Удалить из списка элемент после первого элемента с положительнымзначением. 2. Создать линейный однонаправленный список из целых чисел. Определить среднее арифметическое значений всех элементов списка, кромевторого и третьего.
6	1. Создать линейный однонаправленный список из вещественных чисел. Удалить из списка элемент перед первым элементом со значением 55. 2. Создать линейный однонаправленный список из целых чисел. Вставить в список два числа 111 и 222 перед первым элементом со

	значением 0.
7	1. Создать линейный однонаправленный список из целых чисел. Удалить из списка первый элемент со значением 10. 2. Создать линейный однонаправленный список строк. Удалить из списка первую строку, заканчивающуюся на цифру
8	1. Создать линейный однонаправленный список из вещественных чисел. Вставить в список число 0.99 перед первым элементом со значением 22. 2. Создать линейный однонаправленный список строк. Посчитать общее количество латинских букв во всех строках
9	1. Создать линейный однонаправленный список из целых чисел. Вставить в список число 10 после каждого элемента с отрицательным значением. 2. Создать линейный однонаправленный список из целых чисел. Вставить в список число 12 после первого элемента большего 10
10	1. Создать линейный однонаправленный список из вещественных чисел.Удалить из списка элемент после каждого элемента с положительным значением. 2. Создать линейный однонаправленный список из вещественных чисел.Удалить из списка начальный (первый) элемент и конечный элемент. Учесть возможность пустого списка и списка из одного элемента (тогдарезультат – пустой список).
11	1. Создать линейный однонаправленный список из целых чисел. Определить среднее арифметическое значений элементов списка, начиная с 5-го элемента списка до конца (если элементов меньше пяти, выдатьрезультат 0). 2. Создать линейный однонаправленный список из целых чисел. Удалить из списка два последних элемента. Учесть возможность пустого спискаи списка из одного элемента (тогда результат – пустой список).
12	1. Создать линейный однонаправленный список из целых чисел. Удалить из списка первый элемент со значением 10. 2. Создать линейный однонаправленный список строк. Удалить первую строку-дубликат (совпадающую с предыдущей строкой).
13	1. Создать линейный однонаправленный список из вещественных чисел. Вставить в список число 0.99 перед каждым элементом со значением 55. 2. Создать линейный однонаправленный список из целых чисел. Вставить в список число 12 перед первым элементом, равным 7.
14	1. Создать линейный однонаправленный список из вещественных чисел. Удалить из списка элемент перед положительным первым элементом. 2. Создать линейный однонаправленный список из вещественных чисел.Удалить из списка первый элемент, равный 10. Учесть возможность пустого списка и списка из одного элемента (тогда результат

	– пустой список).
15	1. Создать линейный однонаправленный список из целых чисел. Удалить из списка первый элемент со значением меньше -10. 2. Создать линейный однонаправленный список из символов. Удалить из списка элемент, после символа @. Учесть возможность пустого списка и списка из одного элемента (тогда результат – пустой список).

Средний уровень

Задание 5: написать программу в соответствии с индивидуальным зада-

№ вар.	Задача
1	1. Создать линейный однонаправленный список из вещественных чисел. Удалить из списка элемент перед каждым элементом со значением 55. 2. Создать линейный двунаправленный список из целых чисел. Вставить в список число 11 после каждого элемента, равного 9.
2	1. Создать линейный двунаправленный список из вещественных чисел. Удалить из списка элемент после каждого элемента с отрицательнымзначением. 2. Создать линейный однонаправленный список из вещественных чисел. Удалить из списка элемент перед каждым элементом со значением винтервале от 10 до 20.
3	1. Создать линейный однонаправленный список из вещественных чисел.Вставить в список число 1.5 после каждого элемента с отрицательным значением. 2. Создать линейный двунаправленный список из символов. Удалить из списка элемент после каждого символа &.
4	1. Создать линейный однонаправленный список из вещественных чисел.Определить сумму элементов списка со значениями больше либо равными 15. 2. Создать линейный однонаправленный список из вещественных чисел. Вставить в список число 13.5 после первого элемента со значениембольшим 2.
5	1. Создать линейный однонаправленный список из вещественных чисел. Удалить из списка первый элемент меньший по модулю 5. 2. Создать линейный однонаправленный список из вещественных чисел. Определить среднее значение элементов списка со значениями меньшелибо равными 15. Удалить из списка элементы, которые больше 25.
6	1. Создать линейный однонаправленный список из вещественных чисел. Продублировать в списке первый положительный элемент (если такогонет, оставить список без изменения). 2. Создать линейный однонаправленный список из целых чисел. Удалить из списка первый элемент, больший числа 4. Вставить в список число10 перед каждым числом, равным 15.
7	1. Создать линейный однонаправленный список из целых чисел. Удалить из списка первый четный элемент, стоящий на нечетной позиции. 2. Создать линейный однонаправленный список из вещественных чисел. Вставить в список первый отрицательный элемент перед каждымчислом, равным 20 (если таких нет, оставить список без изменения).
8	1. Создать линейный двунаправленный список из целых чисел. Вставить в список число 66 после каждого элемента с отрицательным значением.

	2. Создать линейный однонаправленный список из целых чисел. Удалитьиз списка каждый элемент, кратный трем. Вставить в список число 88 после каждой пары равных рядом стоящих чисел.
9	1. Создать линейный однонаправленный список из вещественных чисел.Удалить из списка элемент перед каждым элементом со значением -2. Вставить число 33 в конец списка. 2. Создать линейный двунаправленный список из целых чисел. Вставить всписок число 25 перед каждым элементом с положительным значением. Удалить из списка все отрицательные числа.
10	1. Создать линейный двунаправленный список из целых чисел. Удалитьиз списка элемент после каждого элемента, равного 4. Вставить число 0 перед каждым числом 1. 2. Создать линейный однонаправленный список из символов. Удалить изсписка элемент перед каждым символом ^. Определить количество символов * в списке.
11	1. Создать линейный однонаправленный список из вещественных чисел.Вставить в список число 1.5 после каждого элемента с отрицательным значением. Удалить из списка все числа от 2 до 5. 2. Создать линейный двунаправленный список из вещественных чисел. Удалить из списка элементы, у которых дробная часть больше 0,5.
12	1. Создать линейный однонаправленный список из целых чисел. Определить сумму элементов списка со значениями больше либо равными 15.Удалить из списка все значения, которые меньше 5. 2. Создать линейный однонаправленный список из целых чисел. Вставитьв список число 0 перед каждым числом от 2 до 7. Определить сумму чисел больших 7.
13	1. Создать линейный однонаправленный список из символов. Удалить из списка первый элемент, код которого меньше 48. Вставить символ %после каждой цифры. 2. Создать линейный однонаправленный список из вещественных чисел. Определить максимальное из элементов списка и вставить его послекаждого элемента со значением 1.
14	1. Создать линейный однонаправленный список из вещественных чисел.Вставить первый положительный элемент списка после каждого отрицательного числа (если такого нет, оставить список без изменения). 2. Создать линейный однонаправленный список из целых чисел. Удалитьиз списка первый элемент кратный 5. Вставить число 44 перед каждым числом кратным 7.
15	1. Создать линейный однонаправленный список из целых чисел. Вставить в список последний четный элемент после каждого нечетного элемента. 2. Создать линейный однонаправленный список из вещественных чисел.Вычислить среднее значение элементов списка и вставить

	число 11 перед каждым числом, большим среднего значения.

Высокий уровень

Задание 6: написать программу в соответствии с индивидуальным зада-

№ вар.	Задача
1	Создать циклический двунаправленный список из вещественных чисел. Удалить из списка элемент перед каждым элементом со значением 3.
2	Создать циклический двунаправленный список из целых чисел. Удалить из списка первый элемент со значением 10.
3	Создать циклический двунаправленный список из целых чисел. Удалить из списка последний элемент со значением меньшим 15.
4	Создать циклический однонаправленный список из вещественных чисел. Вставить в список число 2.5 перед каждым элементом с положительным значением.
5	Дан указатель P_1 на начало односвязного линейного списка. Преобразовать исходную (*односвязную*) цепочку в *двусвязную*, в которой каждый элемент связан не только с последующим элементом (с помощью поля Next), но и с предыдущим (с помощью поля Prev). Поле Prev первогоэлемента положить равным nil. Вывести на экран преобразованную цепочку в обратном порядке.
6	Дан указатель P_1 на первый элемент непустого двусвязного списка. Продублировать в списке все элементы с нечетными значениями (новые элементы добавлять перед существующими элементами с такими же значениями) и вывести указатель на первый элемент преобразованного списка.
7	Дан указатель P_1 на первый элемент непустого двусвязного списка. Продублировать в списке все элементы с нечетными значениями (новые элементы добавлять после существующих элементов с такими же значениями) и вывести указатель на последний элемент преобразованного списка.
8	Дан указатель P_1 на первый элемент двусвязного списка, содержащего не менее двух элементов. Удалить из списка все элементы с нечетными номерами и вывести указатель на первый элемент преобразованного списка. После удаления элементов из списка освобождать память, которую они занимали
9	Дан указатель P_1 на первый элемент непустого двусвязного списка. Удалить из списка все элементы с нечетными значениями и вывести указатель на первый элемент преобразованного списка (если в результате удаления элементов список окажется пустым, то вывести nil). После удаления элементов из списка освобождать память, которую они занимали

10	Дано число K (> 0) и указатель P_0 на один из элементов непустого двусвязного списка. Переместить в списке данный элемент на K позиций вперед (если после данного элемента находится менее K элементов,то переместить его в конец списка). Вывести указатели на первый и последний элементы преобразованного списка. Операции выделенияи освобождения памяти не использовать, поля с данными (Data) не изменять.
11	Дан указатель P_1 на первый элемент непустого двусвязного списка. Перегруппировать его элементы, переместив все элементы с нечетными номерами в конец списка (в том же порядке) и вывести указатель на первый элемент преобразованного списка. Операции выделения и освобождения памяти не использовать, поля с данными (Data) не изменять.
12	Даны два непустых двусвязных списка и связанные с ними указатели: P_A и P_B указывают на первый и последний элементы первого списка, P_C — на один из элементов второго. Объединить исходные списки, поместив все элементы первого списка (в том же порядке) после данного элемента второго списка, и вывести указатели на первый и последний элементы объединенного списка. Операции выделения и освобождения памяти не использовать
13	Даны указатели P_X и P_Y на два различных элемента двусвязного списка; элемент с адресом P_X находится в списке перед элементом с адресом P_Y, но не обязательно рядом с ним. Переместить элементы, расположенные между данными элементами (не включая данные элементы) в новый список (в том же порядке). Вывести указатели на первые элементы преобразованного и нового списков. Если новый список окажется пустым, то связанный с ним указатель положить равным nil. Операции выделения и освобождения памяти не использовать.
14	Даны указатели P_1 и P_2 на первый и последний элементы непустого двусвязного списка, содержащего четное количество элементов. Преобразовать список в два *циклических* списка, первый из которых содержит первую половину элементов исходного списка, а второй — вторую половину. Вывести указатели P_A и P_B на два средних элемента исходного списка (элемент с адресом P_A должен входить в первый циклический список, а элемент с адресом P_B — во второй). Операции выделенияи освобождения памяти не использовать.

15	Дано число K (> 0) и указатели P_1 и P_2 на первый и последний элементы непустого двусвязного списка. Осуществить *циклический сдвиг* элементов списка на K позиций вперед (то есть в направлении от начала к концу списка) и вывести указатели на первый и последний элементы полученного списка. Для выполнения циклического сдвига преобразовать исходный список в циклический, после чего «разорвать» его в позиции, соответствующей данному значению K. Операции выделения и освобождения памяти не использовать.
16	Даны указатели P_1, P_2 и P_3 на первый, последний и текущий элементы двусвязного списка (если список является пустым, то $P_1 = P_2 = P_3$ = nil). Также дано число N (> 0) и набор из N чисел. Описать тип TList – записьс полями First, Last и Current типа PNode (поля указывают соответствен- но на *первый*, *последний* и *текущий* элементы списка) — и процедуру InsertLast(L, D), которая добавляет новый элемент со значением Dв конец списка L (L – входной и выходной параметр типа TList, D – входной параметр целого типа). Добавленный элемент становится текущим. С помощью этой процедуры добавить в конец исходного списка данный набор чисел (в том же порядке) и вывести новые адреса его первого, последнего и текущего элементов.
17	Даны указатели P_1, P_2 и P_3 на первый, последний и текущий элементы двусвязного списка (если список является пустым, то $P_1 = P_2 = P_3$ = nil). Также дано число N (> 0) и набор из N чисел. Используя тип TList (см. задание 16), описать процедуру InsertFirst(L, D), которая добавляет новый элемент со значением D в начало списка L (L – входной и выходной параметр типа TList, D – входной параметр целого типа). Добавленный элемент становится текущим. С помощью этой процедуры добавить в начало исходного списка данный набор чисел (добавленные числа будут располагаться в списке в обратном порядке) и вывести новые адреса его первого, последнего и текущего элементов.
18	Дан непустой двусвязный список, первый, последний и текущий элементы которого имеют адреса P_1, P_2 и P_3. Также даны пять чисел. Используя тип TList (см. задание 16), описать процедуру InsertBefore(L, D), которая вставляет новый элемент со значением D перед текущим элементом списка L (L — входной и выходной параметр типа TList, D — входной параметр целого типа). Вставленный элемент становится текущим.С помощью этой процедуры вставить пять данных чисел в исходный список и вывести новые адреса его первого, последнего и текущего элементов.
19	Дан непустой двусвязный список, первый, последний и текущий элементы которого имеют адреса P_1, P_2 и P_3. Также даны пять чисел. Используя тип TList (см. задание 16), описать процедуру InsertAfter(L, D), которая вставляет новый элемент со значением D после текущего элемента списка L (L – входной и выходной параметр типа TList, D – входной параметр целого типа). Вставленный элемент становится текущим.С помощью этой процедуры вставить пять данных чисел в исходный

	список и вывести новые адреса его первого, последнего и текущего элементов
20	Создать циклический двунаправленный список из целых чисел. Определить произведение чётных значений элементов списка и вставить элемент со значением, равным вычисленному произведению в конец списка.