Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №6

По дисциплине «Основы алгоритмизации и программирования»

За 2 семестр

Тема: «Динамические структуры данных»

Выполнили:

Студенты 1 курса

Группы ИИ-21(1)

Кирилович А. А.

Проверил:

Гирель Т. Н.

Брест 2022

**Цель работы:** приобретение навыков работы с динамической памятью и указателями на C++. Изучение принципов работы с динамическими структурами данных: списками и деревьями.

**Ход работы:**

**Вариант 4**

**Задание 1**

Напишите программу, содержащую процедуру, которая каждый отрицательный элемент дерева заменяет на положительный, а положительный превращает в ноль.

**Код:**

#include <iostream>

#include <string>

#include <stdlib.h>

using namespace std;

bool is\_left;

int tabs = 0;

struct Node {

    int val;

    Node\* left;

    Node\* right;

    Node(int \_val) : val(\_val), left(nullptr), right(nullptr) {}

};

struct tree {

    Node\* root;

    tree() : root(nullptr) {}

    bool is\_empty() {

        return root == nullptr;

    }

    void push\_back(int \_val, Node\* minor\_root) {

        if (is\_empty()) {

            Node\* p = new Node(\_val);

            root = p;

            return;

        }

        Node\* q = minor\_root;

        if (q->left == nullptr) {

            Node\* p = new Node(\_val);

            q->left = p;

            is\_left = true;

            return;

        }

        if (q->right == nullptr) {

            Node\* p = new Node(\_val);

            q->right = p;

            is\_left = false;

            return;

        }

        if (is\_left) {

            push\_back(\_val, q->right);

        }

        else {

            push\_back(\_val, q->left);

        }

    }

    void print(Node\* minor\_root) {

        if (!minor\_root) return;

        tabs++;

        print(minor\_root->left);

        for (int i = 0; i < tabs; i++) cout << " ";

        cout << minor\_root->val << endl;

        print(minor\_root->right);

        tabs--;

        return;

    }

    void task1(Node\* minor\_root) {

        if (!minor\_root) return;

        task1(minor\_root->left);

        if (minor\_root->val < 0) {

            minor\_root->val = abs(minor\_root->val);

        }

        else {

            minor\_root->val = 0;

        }

        task1(minor\_root->right);

        return;

    }

    void free\_tree(Node\* minor\_root) {

        if (!minor\_root) return;

        free\_tree(minor\_root->left);

        free\_tree(minor\_root->right);

        delete minor\_root;

        return;

    }

};

int main() {

    tree tr;

    string data;

    cout << "Enter the values you need to write to the list. To stop recording write 'q':" << endl;

    while (1) {

        cin >> data;

        if (data == "q") {

            break;

        }

        try {

           tr.push\_back(atoi(data.c\_str()), tr.root);

        }

        catch(const std::exception& e) {

            std::cerr << e.what() << '\n';

        }

    }

    cout << "\n\n";

    tr.print(tr.root);

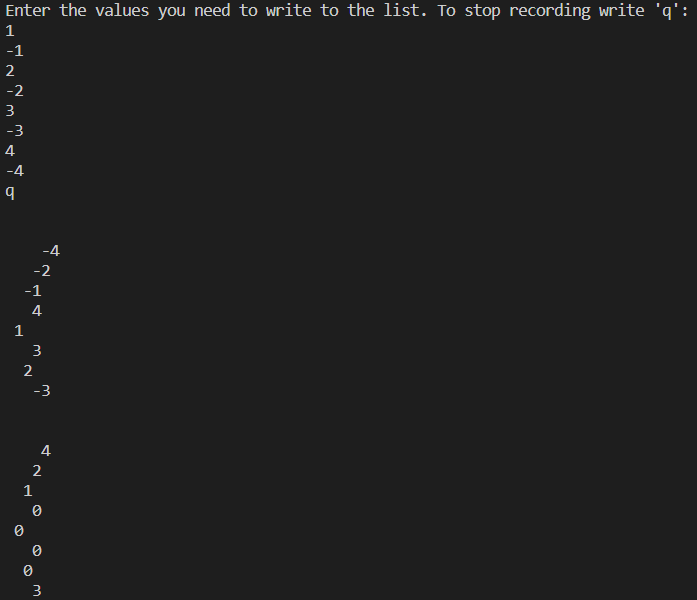
    tr.task1(tr.root);

    cout << "\n\n";

    tr.print(tr.root);

    tr.free\_tree(tr.root);

}



**Задание 2**

Создать текстовые файлы, содержащие один текстовую, а другой числовую информацию (количество слов и чисел должно быть одинаковым). Используя стек, создать другой текстовый файл, в котором числа и слова чередовались и были бы записаны в обратном порядке.

**Код:**

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

struct Node {

    string val;

    Node\* next;

    Node(string \_val) : val(\_val), next(nullptr) {}

};

struct stack {

    Node\* first;

    stack() : first(nullptr) {}

    bool is\_empty() {

        return first == nullptr;

    }

    void add(string \_valToAdd) {

        Node\* p = new Node(\_valToAdd);

        if (is\_empty()) {

            first = p;

            return;

        }

        p->next = first;

        first = p;

    }

    ~stack() {

        while (first != NULL) {

            Node \*temp = first->next;

            delete first;

            first = temp;

        }

    }

};

int main() {

    stack st;

    string PATH\_words = "file2\_words.txt";

    string PATH\_numbers = "file2\_numbers.txt";

    string PATH\_stack = "file2\_stack.txt";

    ofstream file1\_out;

    ofstream file2\_out;

    ofstream file\_stack\_out;

    ifstream file1\_in;

    ifstream file2\_in;

    file1\_out.open(PATH\_words, ios\_base::out);

    file1\_out << "One\nTwo\nThree\nFour\nFive\nSix\nSeven\nEight\nNine\nTen";

    file1\_out.close();

    file2\_out.open(PATH\_numbers, ios\_base::out);

    file2\_out << "1\n2\n3\n4\n5\n6\n7\n8\n9\n10";

    file2\_out.close();

    file1\_in.open(PATH\_words, ios\_base::in);

    file2\_in.open(PATH\_numbers, ios\_base::in);

    string s1;

    string s2;

    while (getline(file1\_in, s1) && getline(file2\_in, s2)) {

        st.add(s1);

        st.add(s2);

    }

    file1\_in.close();

    file2\_in.close();

    file\_stack\_out.open(PATH\_stack, ios\_base::out);

    Node\* p = st.first;

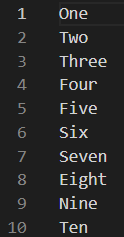
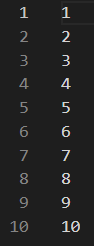
    while (p) {

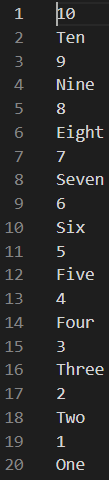
        file\_stack\_out << p->val << "\n";

        p = p->next;

    }

}



**Задание 3**

Написать программу, содержащую процедуры формирования и просмотра списка и функцию, подсчитывающую количество слов списка, которые начинаются той же литерой что и следующее слово.

**Код:**

#include <iostream>

#include <string>

using namespace std;

struct Node {

    string val;

    Node\* next;

    Node(string \_val) : val(\_val), next(nullptr) {}

};

struct list {

    Node\* first;

    Node\* last;

    list() : first(nullptr), last(nullptr) {}

    bool is\_empty() {

        return first == nullptr;

    }

    void push\_back(string \_val) {

        Node\* p = new Node(\_val);

        if (is\_empty()) {

            first = p;

            last = p;

            return;

        }

        last->next = p;

        last = p;

    }

    void print() {

        if (is\_empty()) return;

        Node\* p = first;

        cout << "List: ";

        while (p) {

            cout << p->val << " ";

            p = p->next;

        }

        cout << endl;

    }

    Node\* operator[] (const int index) {

        if (is\_empty()) return nullptr;

        Node\* p = first;

        for (int i = 0; i < index; i++) {

            p = p->next;

            if (!p) return nullptr;

        }

        return p;

    }

    ~list() {

        while (first != NULL) {

            Node \*temp = first->next;

            delete first;

            first = temp;

        }

    }

};

void task3(list l) {

    int i = 0, k = 0;

    while (l[i]->next != nullptr) {

        if (l[i]->val == l[i + 1]->val) {

            k++;

        }

        i++;

    }

    cout << "The number of words in the list that begin with the same letter as the next word: " << k << endl;

}

int main() {

    list l;

    string data;

    cout << "Enter the values you need to write to the list. To stop recording write 'q':" << endl;

    while (1) {

        cin >> data;

        if (data == "q") {

            break;

        }

        l.push\_back(data);

    }

    l.print();

    if (l.is\_empty()) {

        cout << "The list is empty." << endl;

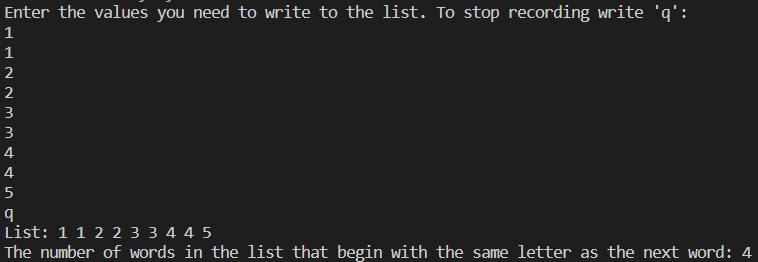
    }

    else {

        task3(l);

    }

}

****

**Задание 4**

Написать программу, содержащую процедуру, которая вставляет новый элемент перед каждым вхождением заданного элемента. Если элементы не найдены, то выдать на экран соответствующие сообщение.

**Код:**

#include <iostream>

#include <string>

using namespace std;

struct Node {

    string val;

    Node\* next;

    Node(string \_val) : val(\_val), next(nullptr) {}

};

struct list {

    Node\* first;

    Node\* last;

    list() : first(nullptr), last(nullptr) {}

    bool is\_empty() {

        return first == nullptr;

    }

    void push\_back(string \_val) {

        Node\* p = new Node(\_val);

        if (is\_empty()) {

            first = p;

            last = p;

            return;

        }

        last->next = p;

        last = p;

    }

    void print() {

        if (is\_empty()) return;

        Node\* p = first;

        cout << "List: ";

        while (p) {

            cout << p->val << " ";

            p = p->next;

        }

        cout << endl;

    }

    void add\_first(string \_valToAdd) {

        if (is\_empty()) return;

        Node\* p = new Node(\_valToAdd);

        p->next = first;

        first = p;

    }

    Node\* operator[] (const int index) {

        if (is\_empty()) return nullptr;

        Node\* p = first;

        for (int i = 0; i < index; i++) {

            p = p->next;

            if (!p) return nullptr;

        }

        return p;

    }

    ~list() {

        while (first != NULL) {

            Node \*temp = first->next;

            delete first;

            first = temp;

        }

    }

};

int main() {

    list l;

    string dataLabel, data;

    cout << "Enter the values you need to write to the list. To stop recording write 'q':" << endl;

    while (1) {

        cin >> data;

        if (data == "q") {

            break;

        }

        l.push\_back(data);

    }

    l.print();

    if (l.is\_empty()) {

        cout << "The list is empty." << endl;

    }

    else {

        cout << "\nEnter the value of the element before which you want to add the new element." << endl;

        cin >> dataLabel;

        cout << "Enter the value of the element you want to add." << endl;

        cin >> data;

        int  i = 0;

        if (l[0]->val == dataLabel) {

            l.add\_first(data);

            i = 2;

        }

        while (l[i] != nullptr) {

            if (l[i]->val == dataLabel) {

                Node\* p = new Node(data);

                Node\* buffer = new Node(dataLabel);

                buffer->next = l[i - 1]->next;

                l[i - 1]->next = p;

                p->next = buffer->next;

                delete buffer;

                i++;

            }

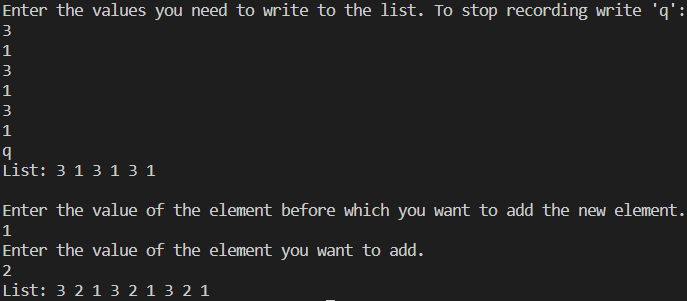
            i++;

        }

        l.print();

    }

}



**Задание 5**

Написать программу, содержащую процедуру, которая удаляет из списка М за каждым вхождением элемента Е один элемент, если такой есть и он отличен от Е.

**Код:**

#include <iostream>

#include <string>

using namespace std;

struct Node {

    string val;

    Node\* next;

    Node(string \_val) : val(\_val), next(nullptr) {}

};

struct list {

    Node\* first;

    Node\* last;

    list() : first(nullptr), last(nullptr) {}

    bool is\_empty() {

        return first == nullptr;

    }

    void push\_back(string \_val) {

        Node\* p = new Node(\_val);

        if (is\_empty()) {

            first = p;

            last = p;

            return;

        }

        last->next = p;

        last = p;

    }

    void print() {

        if (is\_empty()) {

            cout << "List is empty.";

            return;

        }

        Node\* p = first;

        cout << "List: ";

        while (p) {

            cout << p->val << " ";

            p = p->next;

        }

        cout << endl;

    }

    void remove\_first() {

        if (is\_empty()) return;

        Node\* p = first;

        first = p->next;

        delete p;

    }

    Node\* operator[] (const int index) {

        if (is\_empty()) return nullptr;

        Node\* p = first;

        for (int i = 0; i < index; i++) {

            p = p->next;

            if (!p) return nullptr;

        }

        return p;

    }

    ~list() {

        while (first != NULL) {

            Node \*temp = first->next;

            delete first;

            first = temp;

        }

    }

};

int main () {

    list l;

    string data;

    cout << "Enter the values you need to write to the list. To stop recording write 'q':" << endl;

    while (1) {

        cin >> data;

        if (data == "q") {

            break;

        }

        l.push\_back(data);

    }

    l.print();

    cout << "Enter the value of the element before which you want to remove the element." << endl;

    cin >> data;

    if (l.is\_empty()) {

        cout << "The list is empty." << endl;

    }

    else {

        int  i = 0;

        if (l[1]->val == data) {

            if (l[1] != l[0]) {

                l.remove\_first();

                i = 2;

            }

        }

        while (l[i] != nullptr) {

            if (l[i]->val == data) {

                if (l[i - 1]->val != l[i]->val) {

                    Node\* p = l[i - 1];

                    l[i - 2]->next = l[i];

                    delete p;

                }

            }

            else {

                i++;

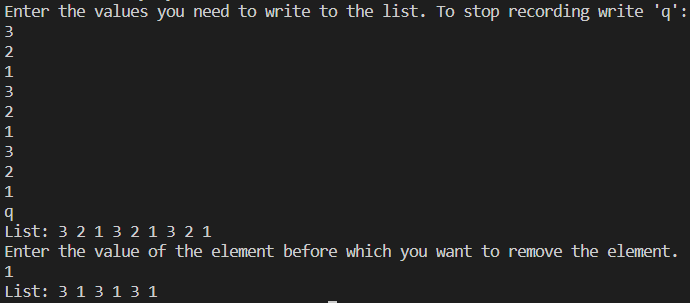
            }

        }

    }

    l.print();

}

****

**Вывод:** приобрел навыки работы с динамической памятью и указателями на C. Изучил принципы работы с динамическими структурами данных: списками и деревьями.