

Федеральное государственное бюджетное образовательное  
учреждение высшего образования  
«Сибирский государственный университет телекоммуникаций и  
информатики»

Кафедра ПМиК

КУРСОВАЯ РАБОТА  
по дисциплине  
«Объектно-ориентированное программирование»  
Тема: «Англо-русский словарь с использованием AVL-дерева»

Выполнил:  
студент группы ИП-213  
Дмитриев А. А.  
Проверил:  
Преподаватель  
Кафедры ПМиК  
Дементьева К. И.

Новосибирск 2023

## СОДЕРЖАНИЕ

1. Постановка задачи
2. Технологии ООП
3. Структура классов
4. Программная реализация
5. Результаты работы
6. Заключение
7. Приложение

## 1. ПОСТАНОВКА ЗАДАЧИ

Написать программу, используя объектно-ориентированный подход. Иерархия классов должна включать минимум четыре класса, один из которых – абстрактный. Язык и среда программирования – C++. Англо-русский словарь с использованием AVL-дерева. Дерево должно быть описано как потомок объекта Двоичное дерево. В качестве методов должны быть описаны функция вычисления высоты, поиск заданного элемента, вставка и удаление узла.

## 2. ТЕХНОЛОГИИ ООП

Для реализации программного продукта использовались такие технологии ООП как:

- Абстрактные классы
- Виртуальные методы
- Конструкторы
- Деструкторы
- Наследование
- Инкапсуляция
- Списки инициализации

## 3. СТРУКТУРА КЛАССОВ

```
class BinaryTree {  
public:  
    virtual void insert(const string& key, const string& value) = 0;  
    virtual void remove(const string& key) = 0;  
    virtual string search(const string& key) const = 0;  
    virtual int height() const = 0;  
};
```

Класс BinaryTree является абстрактным и содержит в себе методы нужные для построения дерева, подсчёта его высоты, и поиска по нему.

```
class DictionaryNode {  
public:  
    string key;  
    string value;  
    int height;
```

```
DictionaryNode* left;  
DictionaryNode* right;
```

```
DictionaryNode(const string& k, const string& v) : key(k), value(v), height(1),  
left(nullptr), right(nullptr) {}  
};
```

Класс DictionaryNode отвечает за хранение слова либо на русском языке, либо английского слова, и его перевода.

Также этот класс содержит список инициализации.

```
class AVLTree : public BinaryTree {  
    ...  
}
```

Класс AVLTree отвечает за создание АВЛ-дерева, наследует класс BinaryTree.

```
class Dictionary {  
    ...  
}
```

В классе Dictionary создаются два словаря: русско-английский и англо-русский в виде объектов russianTree и englishTree класса AVLTree.

## 4. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

```
#include <iostream>  
#include <fstream>  
#include <string>  
#include <algorithm>
```

```
using namespace std;
```

```
class BinaryTree {  
public:
```

```

virtual void insert(const string& key, const string& value) = 0;
virtual void remove(const string& key) = 0;
virtual string search(const string& key) const = 0;
virtual int height() const = 0;
};

```

```

class DictionaryNode {
public:
    string key;
    string value;
    int height;
    DictionaryNode* left;
    DictionaryNode* right;

```

```

    DictionaryNode(const string& k, const string& v) : key(k), value(v), height(1),
    left(nullptr), right(nullptr) {}
};

```

```

class AVLTree : public BinaryTree {
private:
    DictionaryNode* root;

    int getHeight(DictionaryNode* node) const {
        return node ? node->height : 0;
    }

    int getBalance(DictionaryNode* node) const {
        return node ? getHeight(node->left) - getHeight(node->right) : 0;
    }

    DictionaryNode* rotateRight(DictionaryNode* y) {
        DictionaryNode* x = y->left;
        DictionaryNode* T2 = x->right;

        x->right = y;

```

```

y->left = T2;

y->height = 1 + max(getHeight(y->left), getHeight(y->right));
x->height = 1 + max(getHeight(x->left), getHeight(x->right));

return x;
}

```

```

DictionaryNode* rotateLeft(DictionaryNode* x) {
    DictionaryNode* y = x->right;
    DictionaryNode* T2 = y->left;

    y->left = x;
    x->right = T2;

    x->height = 1 + max(getHeight(x->left), getHeight(x->right));
    y->height = 1 + max(getHeight(y->left), getHeight(y->right));

    return y;
}

```

```

DictionaryNode* balance(DictionaryNode* node) {
    if (node == nullptr)
        return nullptr;

    node->height = 1 + max(getHeight(node->left), getHeight(node->right));

    int balance = getBalance(node);

    if (balance > 1) {
        if (getBalance(node->left) < 0)
            node->left = rotateLeft(node->left);
        return rotateRight(node);
    }
}

```

```

    if (balance < -1) {
        if (getBalance(node->right) > 0)
            node->right = rotateRight(node->right);
        return rotateLeft(node);
    }

    return node;
}

DictionaryNode* insert(DictionaryNode* node, const string& key, const string& value) {
    if (node == nullptr)
        return new DictionaryNode(key, value);

    if (key < node->key)
        node->left = insert(node->left, key, value);
    else if (key > node->key)
        node->right = insert(node->right, key, value);
    else
        node->value = value;

    return balance(node);
}

DictionaryNode* findMin(DictionaryNode* node) {
    while (node->left != nullptr)
        node = node->left;
    return node;
}

DictionaryNode* remove(DictionaryNode* node, const string& key) {
    if (node == nullptr)
        return nullptr;

    if (key < node->key)
        node->left = remove(node->left, key);

```

```

else if (key > node->key)
    node->right = remove(node->right, key);
else {
    if (node->left == nullptr || node->right == nullptr) {
        DictionaryNode* temp = node->left ? node->left : node->right;
        if (temp == nullptr) {
            temp = node;
            node = nullptr;
        } else {
            *node = *temp;
        }
        delete temp;
    } else {
        DictionaryNode* temp = findMin(node->right);
        node->key = temp->key;
        node->value = temp->value;
        node->right = remove(node->right, temp->key);
    }
}

if (node == nullptr)
    return nullptr;

return balance(node);
}

string search(DictionaryNode* node, const string& key) const {
    if (node == nullptr)
        return "Слово не найдено.";

    if (key < node->key)
        return search(node->left, key);
    else if (key > node->key)
        return search(node->right, key);
    else

```



```

        return node->value;
    }

void print(DictionaryNode* node) const {
    if (node == nullptr)
        return;

    print(node->left);
    cout << node->key << " - " << node->value << endl;
    print(node->right);
}

public:
    AVLTree() : root(nullptr) {}

    void insert(const string& key, const string& value) override {
        root = insert(root, key, value);
    }

    void remove(const string& key) override {
        root = remove(root, key);
    }

    string search(const string& key) const override {
        return search(root, key);
    }

    int height() const override {
        return getHeight(root);
    }

    void print() const {
        print(root);
    }

    ~AVLTree() {}

```

```
};
```

```
class Dictionary {
```

```
private:
```

```
    AVLTree russianTree;
```

```
    AVLTree englishTree;
```

```
public:
```

```
    Dictionary() {}
```

```
    ~Dictionary() {}
```

```
    void loadFromFile(const string& filename) {
```

```
        ifstream file(filename);
```

```
        if (!file.is_open()) {
```

```
            cerr << "Ошибка открытия файла: " << filename << endl;
```

```
            return;
```

```
        }
```

```
        string russianWord, englishWord;
```

```
        while (file >> russianWord >> englishWord) {
```

```
            russianTree.insert(russianWord, englishWord);
```

```
            englishTree.insert(englishWord, russianWord);
```

```
        }
```

```
        file.close();
```

```
    }
```

```
    void interactiveSearch() {
```

```
        string input;
```

```
        while (true) {
```

```
            cout << "Введите слово ('выход' или 'exit' для выхода): ";
```

```
            getline(cin, input);
```

```
            if ((input == "выход" ) || (input == "exit"))
```

```

        break;

    string russianResult = russianTree.search(input);
    if (russianResult != "Слово не найдено.") {
        cout << "Перевод: " << russianResult << endl;
    } else {
        string englishResult = englishTree.search(input);
        if (englishResult != "Слово не найдено.") {
            cout << "Перевод: " << englishResult << endl;
        } else {
            cout << "Слово не найдено." << endl;
        }
    }
}

int height() const {
    cout << "Высота русского дерева: " << russianTree.height() << endl;
    cout << "Высота английского дерева: " << englishTree.height() << endl;
}

void printTree() const {
    cout << "Русское дерево:" << endl;
    russianTree.print();

    cout << endl << "Английское дерево:" << endl;
    englishTree.print();
}

};

int main() {
    Dictionary dictionary;
    dictionary.loadFromFile("data.txt");

    dictionary.height();

```

```

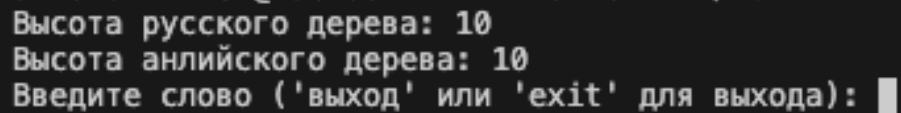
dictionary.interactiveSearch();

cout << "Посмотреть деревья да/нет > ";
string input;
getline(cin, input);
if (input == "да")
    dictionary.printTree();
else
    cout << "Спасибо за просмотр!";
return 0;
}

```

## 5. РЕЗУЛЬТАТЫ РАБОТЫ

При запуске программы выводятся высоты деревьев, хранящих словарь, и пользователя просят ввести слово, процесс нахождения перевода которого будет производиться.

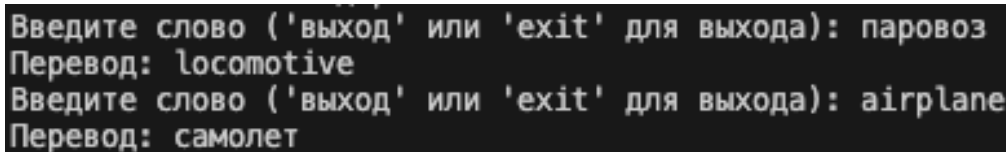


```

Высота русского дерева: 10
Высота анлийского дерева: 10
Введите слово ('выход' или 'exit' для выхода): 

```

Программа может может производить поиск, как русского слова, так и английского перевода.

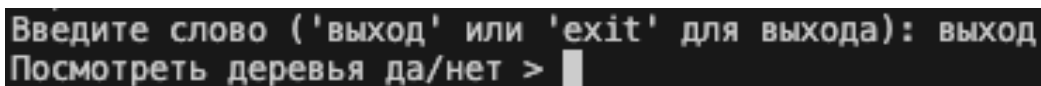


```

Введите слово ('выход' или 'exit' для выхода): паровоз
Перевод: locomotive
Введите слово ('выход' или 'exit' для выхода): airplane
Перевод: самолет

```

Если пользователь захочет выйти, программа запросит возможность просмотра деревьев построенных по данным из файла data.txt



```

Введите слово ('выход' или 'exit' для выхода): выход
Посмотреть деревья да/нет > 

```

```
Посмотреть деревья да/нет > да
Русское дерево:
авангард – avant-garde
август – august
автобус – bus
автограф – autograph
автомобиль – car
авторитет – authority
акула – shark
анархия – anarchy
арена – arena
армия – army
архитектура – architecture
бактерия – bacterium
банан – banana
```

```
эффект – effect
яблоко – apple
язык – language
ящик – box

Английское дерево:
advertisement – реклама
airplane – самолет
anarchy – анархия
apple – яблоко
architecture – архитектура
arena – арена
armchair – кресло
army – армия
art – искусство
assembly – собрание
august – август
```

## 6. ЗАКЛЮЧЕНИЕ

В ходе работы я написал программу используя технологии ООП такие как, абстрактные классы, виртуальные методы, конструкторы, деструкторы, наследование, инкапсуляция, списки инициализации. Закрепил материал изученный на предмете Объектно-Оrientированное программирование.

## 7. ПРИЛОЖЕНИЕ

Файл data.txt

дом house

кошка cat

стол table

окно window  
солнце sun  
рука hand  
глаз eye  
вода water  
дерево tree  
книга book  
яблоко apple  
машина car  
телефон phone  
цветок flower  
звезда star  
трава grass  
нос nose  
голова head  
нога leg  
город city  
песок sand  
птица bird  
музыка music  
зима winter  
лето summer  
снег snow  
гора mountain  
рыба fish  
фото photo  
часы watch  
слово word  
ночь night  
день day  
собака dog  
река river  
язык language  
красный red  
синий blue

желтый yellow  
зеленый green  
человек man  
женщина woman  
ребенок child  
школа school  
учитель teacher  
студент student  
работа work  
игра game  
компьютер computer  
волос hair  
ухо ear  
пальто coat  
планета planet  
зеркало mirror  
кресло chair  
кот cat  
ворота gate  
птичка bird  
песня song  
карта map  
платье dress  
кольцо ring  
ноутбук laptop  
чашка cup  
человек man  
женщина woman  
ребенок child  
школа school  
учитель teacher  
студент student  
работа work  
игра game  
компьютер computer

волос hair  
ухо ear  
пальто coat  
планета planet  
зеркало mirror  
кресло chair  
кот cat  
ворота gate  
птичка bird  
песня song  
карта map  
платье dress  
кольцо ring  
ноутбук laptop  
чашка cup  
космос space  
медведь bear  
робот robot  
свитер sweater  
радуга rainbow  
замок castle  
банан banana  
завтрак breakfast  
праздник holiday  
печь oven  
птицелов scarecrow  
гитара guitar  
фотоаппарат camera  
ночник nightlight  
поезд train  
автобус bus  
газета newspaper  
пальто overcoat  
магазин shop  
комната room



завод factory  
календарь calendar  
конфета candy  
танец dance  
картинка picture  
робот robot  
дождь rain  
молоко milk  
пиво beer  
чемпионат championship  
кресло armchair  
тетрадь notebook  
рюкзак backpack  
компас compass  
пивная beerhouse  
герой hero  
свиток scroll  
лампа lamp  
золото gold  
дракон dragon  
август august  
сентябрь september  
октябрь october  
ноябрь november  
декабрь december  
каникулы vacation  
экран screen  
рубашка shirt  
чемодан suitcase  
принтер printer  
стекло glass  
смайлик smiley  
шутка joke  
танцор dancer  
гитарист guitarist

мышь mouse  
заяц hare  
сова owl  
роза rose  
поросенок piglet  
гриб mushroom  
реклама advertisement  
зоопарк zoo  
скейтборд skateboard  
ящик box  
галактика galaxy  
религия religion  
архитектура architecture  
кафе cafe  
печенье cookie  
помидор tomato  
колесо wheel  
небо sky  
подушка pillow  
сапог boot  
цвет color  
страница page  
чемпион champion  
коробка box  
снеговик snowman  
мышление thinking  
магия magic  
цветочный flower  
часовой watchman  
пара couple  
глобус globe  
дверь door  
океан ocean  
гриль grill  
принцесса princess

коктейль cocktail  
стадион stadium  
гербарий herbarium  
ракета rocket  
танк tank  
гном gnome  
дракон dragon  
замороженный frozen  
жалюзи blinds  
белье lingerie  
градус degree  
трюк trick  
пастель pastel  
губка sponge  
жемчуг pearl  
ферма farm  
папоротник fern  
коридор corridor  
марс mars  
санки sled  
песочница sandbox  
собрание assembly  
профессор professor  
живопись painting  
шахматы chess  
маникюр manicure  
леденец lollipop  
пижама pajama  
акула shark  
тропик tropical  
фонтан fountain  
кресло recliner  
резина rubber  
костюм costume  
рыцарь knight

облако cloud  
завтрак breakfast  
депозит deposit  
косметика cosmetics  
космонавт cosmonaut  
автограф autograph  
зонт umbrella  
стакан glass  
забор fence  
блокнот notepad  
отель hotel  
зеркало mirror  
лимон lemon  
клиент client  
невеста bride  
улитка snail  
котенок kitten  
река river  
волшебник wizard  
крокодил crocodile  
паровоз locomotive  
медицина medicine  
продавец seller  
флаг flag  
камера camera  
черепаха turtle  
петух rooster  
рисунок drawing  
самолет airplane  
комета comet  
трюк trick  
спутник satellite  
берег shore  
флейта flute  
маяк lighthouse

секрет secret  
робот robot  
песок sand  
модель model  
коммуникация communication  
класс class  
монитор monitor  
персона person  
объект object  
бизнес business  
финансы finances  
мечта dream  
футбол football  
гитара guitar  
интернет internet  
сердце heart  
парень guy  
вечерина party  
ресторан restaurant  
библиотека library  
поэзия poetry  
искусство art  
культура culture  
география geography  
физика physics  
химия chemistry  
математика mathematics  
литература literature  
революция revolution  
завод factory  
мастерская workshop  
исследование research  
изобретение invention  
композиция composition  
экология ecology

фотография photography  
поэзия poetry  
кино cinema  
телевизор television  
концерт concert  
эксперимент experiment  
автомобиль car  
велосипед bicycle  
картина painting  
технология technology  
робототехника robotics  
геология geology  
футуризм futurism  
лекция lecture  
инженерия engineering  
политика politics  
психология psychology  
генетика genetics  
философия philosophy  
религия religion  
экономика economy  
энергия energy  
экспорт export  
импорт import  
бюрократия bureaucracy  
парламент parliament  
демократия democracy  
авторитет authority  
полиция police  
революция revolution  
анархия anarchy  
армия army  
стратегия strategy  
глобализация globalization  
реформа reform

революция revolution  
обучение education  
робот robot  
дизайн design  
компьютер computer  
генератор generator  
микроб microbe  
бактерия bacterium  
вирус virus  
эпидемия epidemic  
вакцина vaccine  
загадка riddle  
шедевр masterpiece  
реликвия relic  
миф myth  
легенда legend  
экспедиция expedition  
конференция conference  
экспозиция exposition  
этюд etude  
коллекция collection  
перформанс performance  
рецензия review  
симфония symphony  
революция revolution  
авангард avant-garde  
косметология cosmetology  
защита protection  
пенсия pension  
портфель portfolio  
климат climate  
удача luck  
хобби hobby  
проблема problem  
вино wine

ресурс resource  
магнит magnet  
пауза pause  
режим mode  
реставрация restoration  
ценность value  
корень root  
марафон marathon  
блок block  
реакция reaction  
коммунизм communism  
капитализм capitalism  
эксплорер explorer  
интроверт introvert  
экстраверт extrovert  
протокол protocol  
парламент parliament  
функция function  
революция revolution  
бриллиант diamond  
конфликт conflict  
фактор factor  
школьник schoolboy  
инженер engineer  
программист programmer  
калькулятор calculator  
пианист pianist  
университет university  
карьера career  
гармония harmony  
революция revolution  
театр theatre  
арена arena  
эксплорация exploration  
депрессия depression



фонтан fountain  
физиология physiology  
герметик sealant  
катализатор catalyst  
эффект effect  
модель model  
гигант giant  
контракт contract  
транзит transit  
позитив positive  
негатив negative  
хаос chaos  
электрон electron  
грань edge  
лаборатория laboratory  
модерн modern  
гардероб wardrobe  
резерв reserve  
диагноз diagnosis  
экзамен exam  
звук sound  
код code  
рубеж frontier  
компонент component  
хроника chronicle  
легион legion  
сектор sector  
энергия energy  
революция revolution  
схема scheme  
доминанта dominant  
локомотив locomotive  
нота note  
холод cold  
поток stream

план plan

гарантия guarantee

резонанс resonance

благосостояние prosperity

революция revolution