Федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»

им. В.И. Ульянова (Ленина)»

кафедра Апу

**ОТЧЕТ**

**по лабораторной работе №5**

**«АВЛ-дерево»**

Выполнил: Панфёров М. А.

Группа №3371

Преподаватель: Ряскова Е. Б.

Санкт-Петербург

2024

Постановка задачи

Создайте структуру данных для хранения информации о студентах:

* № студенческого билета,
* ФИО студента,
* Курс,
* Дата зачисления,
* Контактная информация.

Организуйте информацию о студентах в виде АВЛ-дерева поиска, упорядоченного по номеру студенческого билета.

Для АВЛ-дерева должны быть реализованы следующие функции:

* добавление информации о студенте,
* вывод АВЛ-дерева на экран (любым способом доступным для восприятия).
* очистка данных АВЛ-дерева,
* поиск студента по номеру студенческого билета, результат поиска в случае успеха – все сведения о найденном студенте,
* удаление информацию о студенте (по № студенческого билета).

Для взаимодействия с пользователем реализуйте в программе меню, содержащее следующие разделы:

* + Добавление студента
  + Вывод информации о студентах
  + Удаление информации о студентах
  + Поиск студента
  + Удаление студента
  + Выход из программы.

Листинг программы

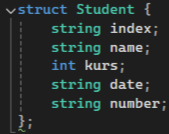


Рис. 1 – Структура студента

Структура студента в себе содержит данные его номера студенческого билета, ФИО, курса, даты зачисления и его контактных данных: можно указать почту или телефон.

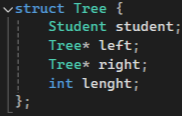


Рис. 2 – Структура дерева

Структура дерева в себе содержит студента, правую и левую ветви дерева и его высоту (подписана как «lenght»).



Рис. 3 – Класс АВЛ-дерева

Класс АВЛ-дерева в себе содержит методы для хранения, распределения по ветвям, вывода, добавления и удаления студентов.

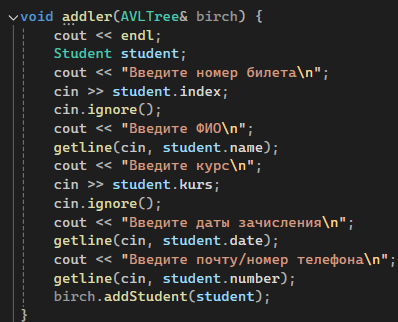


Рис. 4 – Функция добавления студентов

В функцию добаления студентов подаётся АВЛ-дерево. После воодятся даные студента и студент добавляется в дерево.

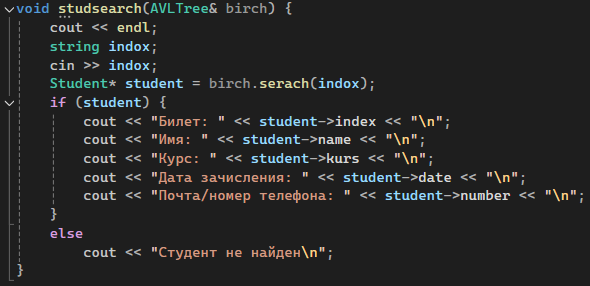


Рис. 5 – Функция поиска студентов

В функцию поиска студентов подаётся АВЛ-дерево. Поиск идёт по индексу, если студент с таким индексом был найден, то его данные будут выведены, иначе программа сообщит, что сиудент не найден.

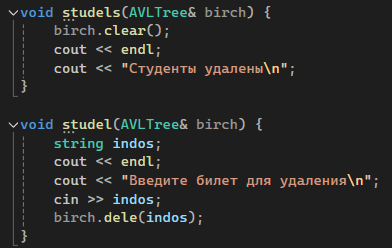


Рис. 6 – Функции удаления студентов и студента

В обе функции удаления подаётся АВЛ-дерево. В studels удаляются все студенты, в studel удаляется студент, при помощи указании его номера билета.

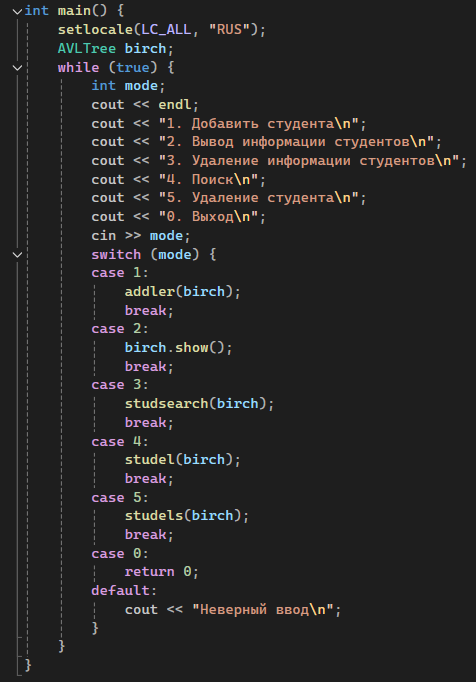


Рис. 7 – Главная структура

В главной функции находится меню, где при вводе определённой цифры будет выполнятся определённое действие.

Код и пример работы программы

AiSD\_Laba5.cpp:

#include "iostream"

#include "string"

using namespace std;

struct Student {

string index;

string name;

int kurs;

string date;

string number;

};

struct Tree {

Student student;

Tree\* left;

Tree\* right;

int lenght;

};

class AVLTree {

private:

Tree\* root;

int lenght(Tree\* leaf) {

return (leaf ? leaf->lenght : 0);

}

int factor(Tree\* leaf) {

return lenght(leaf->right) - lenght(leaf->left);

}

void diflenght(Tree\* leaf) {

int difleft = lenght(leaf->left);

int difright = lenght(leaf->right);

leaf->lenght = (difleft > difright ? difleft : difright);

}

Tree\* turnleft(Tree\* leaf) {

Tree\* newroot = leaf -> right;

leaf->right = newroot->left;

newroot->left = leaf;

diflenght(leaf);

diflenght(newroot);

return newroot;

}

Tree\* turnright(Tree\* leaf) {

Tree\* newroot = leaf->left;

leaf->left = newroot->right;

newroot->right = leaf;

diflenght(leaf);

diflenght(newroot);

return newroot;

}

Tree\* balance(Tree\* leaf) {

diflenght(leaf);

if (factor(leaf) == 2) {

if (factor(leaf->right) < 0)

leaf->right = turnright(leaf->right);

return turnleft(leaf);

}

if (factor(leaf) == -2) {

if (factor(leaf->left) > 0)

leaf->left = turnleft(leaf->left);

return turnright(leaf);

}

return leaf;

}

Tree\* add(Tree\* leaf, Student& student) {

if (!leaf) {

Tree\* newleaf = new Tree{ student, nullptr, nullptr, 1 };

return newleaf;

}

if (student.index < leaf->student.index)

leaf->left = add(leaf->left, student);

else

leaf->right = add(leaf->right, student);

return balance(leaf);

}

void prin(Tree\* leaf) {

if (leaf) {

prin(leaf->left);

cout << endl;

cout << "Билет: " << leaf->student.index << "\n";

cout << "Имя: " << leaf->student.name << "\n";

cout << "Курс: " << leaf->student.kurs << "\n";

cout << "Дата зачисления: " << leaf->student.date << "\n";

cout << "Почта/номер телефона: " << leaf->student.number << "\n";

prin(leaf->right);

}

}

void clear(Tree\* leaf) {

if (leaf) {

clear(leaf->left);

clear(leaf->right);

delete leaf;

}

}

Tree\* finder(Tree\* leaf, const string& index) {

if (!leaf)

return nullptr;

if (index == leaf->student.index)

return leaf;

if (index < leaf->student.index)

return finder(leaf->left, index);

return finder(leaf->right, index);

}

Tree\* delet(Tree\* leaf, const string& index) {

if (!leaf)

return nullptr;

if (index < leaf->student.index)

leaf->left = delet(leaf->left, index);

else if (index > leaf->student.index)

leaf->right = delet(leaf->right, index);

else {

Tree\* left = leaf->left;

Tree\* right = leaf->right;

delete leaf;

if (!right)

return left;

Tree\* min = right;

while (min > left)

min = min->left;

min->right = delet(right, min->student.index);

min->left = left;

return balance(min);

}

return balance(leaf);

}

public:

AVLTree(): root(nullptr){}

~AVLTree() {

clear(root);

}

void addStudent(Student& student) {

root = add(root, student);

}

void show() {

prin(root);

}

void clear() {

clear(root);

root = nullptr;

}

Student\* serach(const string& index) {

Tree\* leaf = finder(root, index);

return leaf ? &leaf->student : nullptr;

}

void dele(const string& index) {

root = delet(root, index);

}

};

void addler(AVLTree& birch) {

cout << endl;

Student student;

cout << "Введите номер билета\n";

cin >> student.index;

cin.ignore();

cout << "Введите ФИО\n";

getline(cin, student.name);

cout << "Введите курс\n";

cin >> student.kurs;

cin.ignore();

cout << "Введите даты зачисления\n";

getline(cin, student.date);

cout << "Введите почту/номер телефона\n";

getline(cin, student.number);

birch.addStudent(student);

}

void studsearch(AVLTree& birch) {

cout << endl;

string indox;

cin >> indox;

Student\* student = birch.serach(indox);

if (student) {

cout << "Билет: " << student->index << "\n";

cout << "Имя: " << student->name << "\n";

cout << "Курс: " << student->kurs << "\n";

cout << "Дата зачисления: " << student->date << "\n";

cout << "Почта/номер телефона: " << student->number << "\n";

}

else

cout << "Студент не найден\n";

}

void studels(AVLTree& birch) {

birch.clear();

cout << endl;

cout << "Студенты удалены\n";

}

void studel(AVLTree& birch) {

string indos;

cout << endl;

cout << "Введите билет для удаления\n";

cin >> indos;

birch.dele(indos);

}

int main() {

setlocale(LC\_ALL, "RUS");

AVLTree birch;

while (true) {

int mode;

cout << endl;

cout << "1. Добавить студента\n";

cout << "2. Вывод информации студентов\n";

cout << "3. Удаление информации студентов\n";

cout << "4. Поиск\n";

cout << "5. Удаление студента\n";

cout << "0. Выход\n";

cin >> mode;

switch (mode) {

case 1:

addler(birch);

break;

case 2:

birch.show();

break;

case 3:

studsearch(birch);

break;

case 4:

studel(birch);

break;

case 5:

studels(birch);

break;

case 0:

return 0;

default:

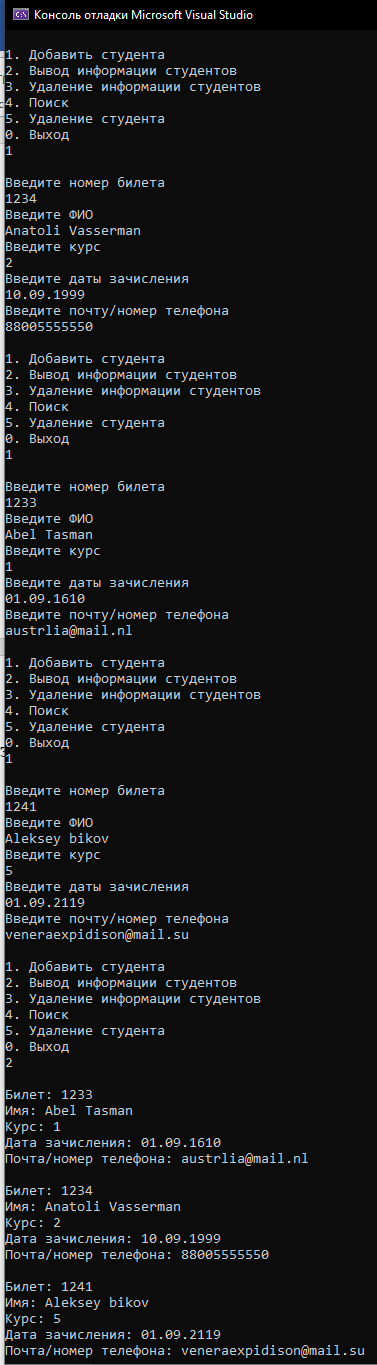
cout << "Неверный ввод\n";

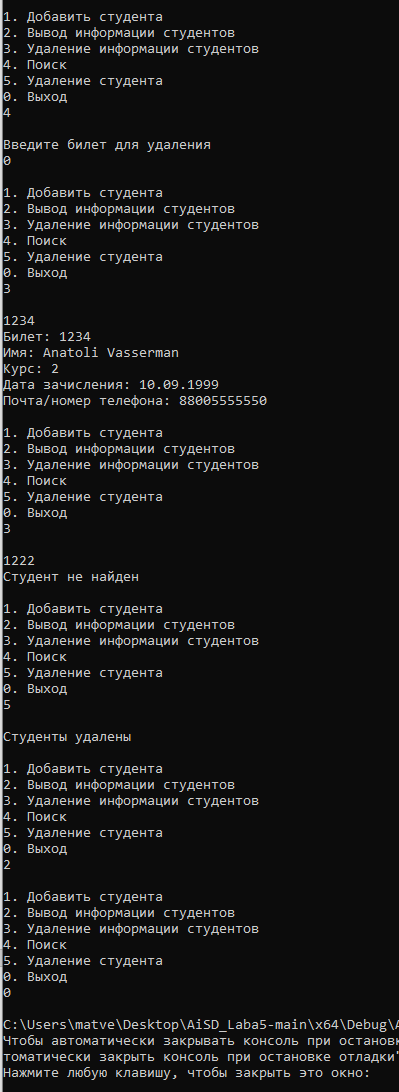
}

}

}

Пример работы программы:





Вывод: я реализовал программу, которая строит АВЛ-дерево, состоящее из студентов. Знаю как оно работает и как им управлять. Теперь могу использовать АВЛ-деревов в своих проектах.