**8 ЛАБОРАТОРНАЯ РАБОТА №8**

**«ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ СПИСКОВ НА ЯЗЫКЕ С/С++»**

**8.1 Цель работы**

Изучение нелинейных структур данных и приобретение навыков разработки и отладки программ, использующих древовидные структуры. Исследование особенностей работы с поисковыми бинарными деревьями на языке С/С++.

**8.2 Вариант задания – 20**

Требуется представить таблицу 8.1 в виде бинарного дерева, элементами которого являются строки таблицы. Написать функции создания и обхода дерева, а также функцию, которая подсчитывает число вершин на n-ом уровне непустого дерева. Значения полей и количество записей в таблице студент выбирает самостоятельно. Программа должна сохранять дерево в файле и создавать его заново при её повторном запуске.

**Таблица *8*.1 – Расписание**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N поезда | Станция отправления | Станция назначения | Время отправления | Время прибытия | Стоимость билета |

**8.3 Порядок выполнения работы**

8.3.1 Разработать алгоритм решения задачи.

8.3.2 Разработать программу на языке Си.

8.3.3 Разработать тестовые примеры. Выполнить тестирование используя разработанные тестовые примеры. Выполнить отладку.

8.3.4 Сделать выводы по проделанной работе.

**8.4 Ход работы**

8.4.1 Для задания был разработан алгоритм решения задачи и отражен на структурных схемах, которые представлены на рисунках 8.1 и 8.2.

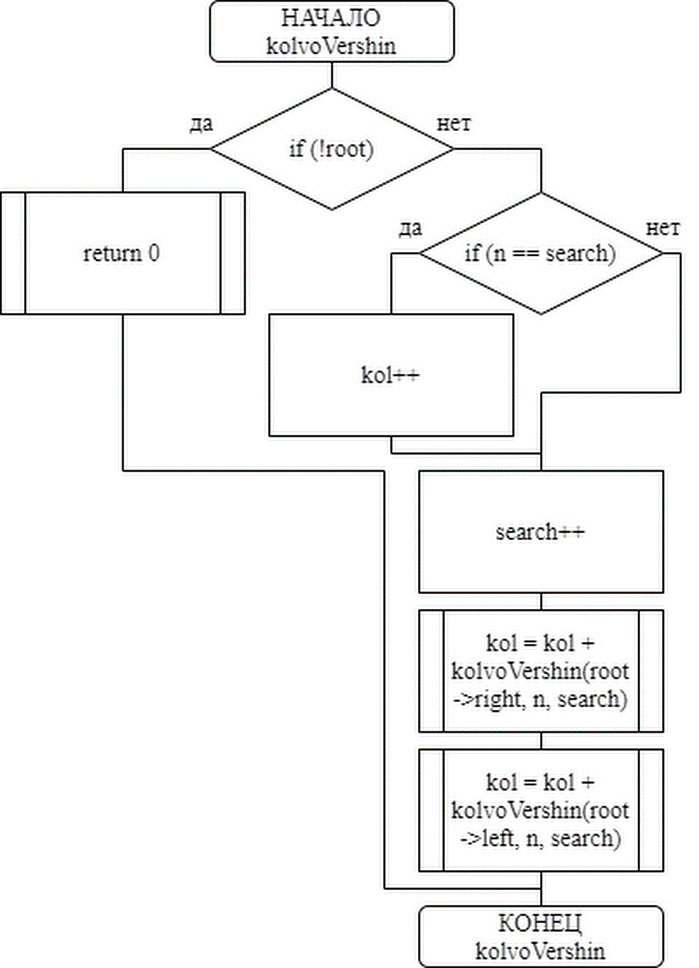
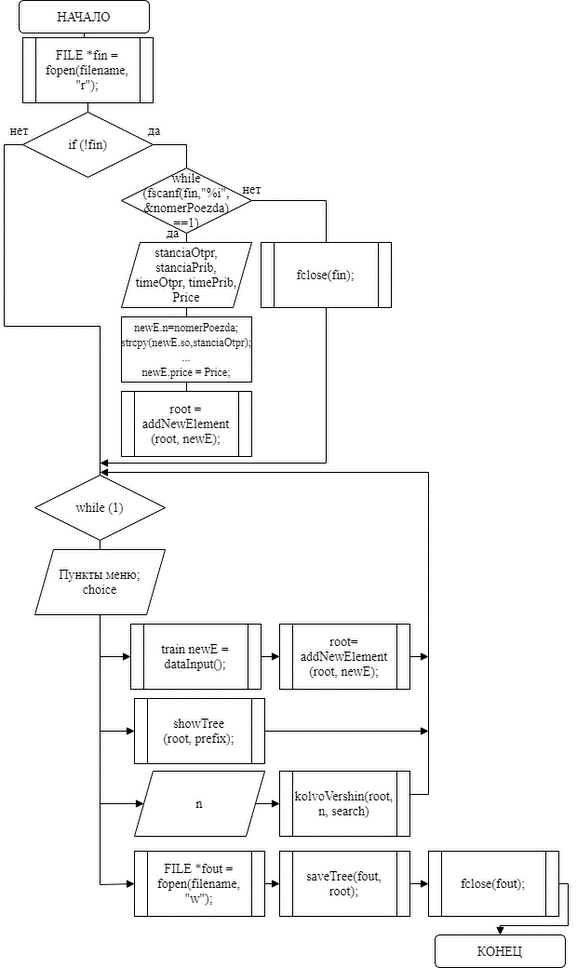


Рисунок 8.1 – Структурная схема основной программы, функции демонстрации дерева, взятия информации об элементе и подстчет количества вершин

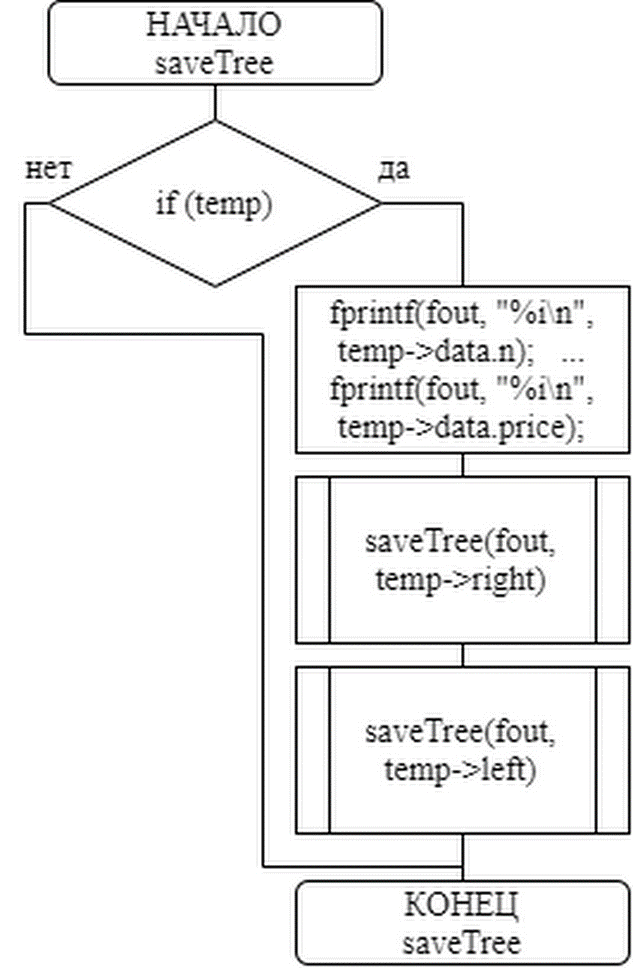
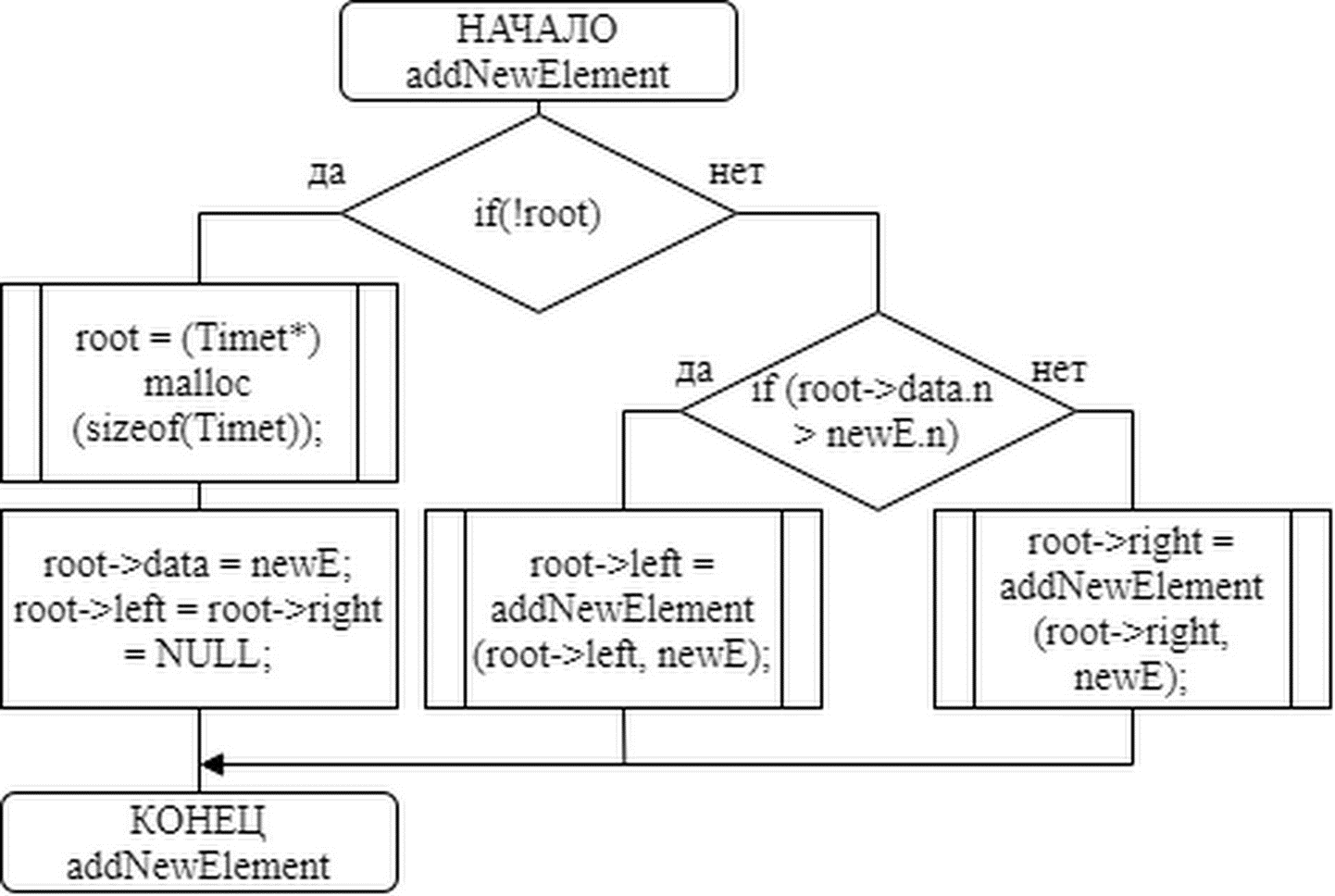


Рисунок 8.2 – Структурная схема функций добавления листа и сохранения дерева

8.4.2 Написана программа на Си согласно вышеописанного алгоритма.

#include <stdio.h>

#include <stdlib.h>

#include <windows.h>

#include <string.h>

//==================Структура с расписанием поездов==================

typedef struct info{

int n; //номер поезда

char so[21], //станция отправления

sp[21], //станция прибытия

vo[6], //время отправления

vp[6]; //время прибытия

int price; //стоимость билета

}train;

typedef struct Timetable{

train data;

struct Timetable \*left;

struct Timetable \*right;

}Timet;

//==================Прототипы функций==================

train dataInput();

Timet \*addNewElement(Timet \*root, const train newE);

void showTree(Timet \*root, int prefix);

int kolvoVershin(Timet \*root, int n, int search);

void saveTree(FILE \*fout, Timet \*root);

void downloadTree(char \*filename, Timet \*\*root);

void goodbye();

//==================Основная программа==================

int main(){

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

char \*filename = "Timetable\_tree.txt";

Timet \*root = NULL;

int choice;

//загрузить дерево из файла

FILE \*fin = fopen(filename, "r");

if (!fin){

printf("Файл с сохранением не удалось прочесть :(\n");

}

else{

train newE;

int nomerPoezda;

while (fscanf(fin, "%i", &nomerPoezda) == 1){

char stanciaOtpr[21];

fscanf(fin, "%s", &stanciaOtpr);

char stanciaPrib[21];

fscanf(fin, "%s", &stanciaPrib);

char timeOtpr[6];

fscanf(fin, "%s", &timeOtpr);

char timePrib[6];

fscanf(fin, "%s", &timePrib);

int Price;

fscanf(fin, "%i", &Price);

newE.n = nomerPoezda;

strcpy(newE.so, stanciaOtpr);

strcpy(newE.sp, stanciaPrib);

strcpy(newE.vo, timeOtpr);

strcpy(newE.vp, timePrib);

newE.price = Price;

root = addNewElement(root, newE);

}

fclose(fin);

}

if (root)

printf("Дерево успешно загружено\n");

else

printf("Дерево не загружено\n");

while(1) {

printf("\n\nВведите любое число для продолжения:\n");

scanf("%i", &choice);

system("cls");

printf("\nВыберите дальнейшее действие: \n");

printf(" 1 - Добавить элемент в дерево; \n");

printf(" 2 - Отобразить дерево; \n");

printf(" 3 - Подсчитать число вершин на n-ом уровне дерева; \n");

printf(" 4 - Сохранить и выйти; \n");

printf(" -> ");

scanf("%i", &choice);

printf("\n");

switch (choice) {

case 1:

{

train newE = dataInput();

root = addNewElement(root, newE);

break;

}

case 2:

{

if (!root){

printf("Дерево отсутствует\n");

break;

}

else{

int prefix = 0;

showTree(root, prefix);

break;

}

}

case 3:

{

int n, search=1;

printf("Функция подсчитывает количество вершин на n-ом уровне дерева\n");

printf("Корень дерева считается первым уровнем\n");

printf("Введите n:"); scanf("%i", &n);

printf("На уровне %i: %i вершин\n", n, kolvoVershin(root, n, search));

break;

}

case 4:

{

FILE \*fout = fopen(filename, "w");

if (!fout)

{

printf("Файл с сохранением не создан :(\n");

goodbye();

return 0;

}

saveTree(fout, root);

fclose(fout);

goodbye();

return 0;

}

default:

printf("Вы ввели неверный символ\n");

break;

}

}

}

//==================Добавить элемент в дерево==================

Timet \*addNewElement(Timet \*root, const train newE){

if(!root){

root = (Timet\*)malloc(sizeof(Timet));

root->data = newE;

root->left = root->right = NULL;

}

else{

if (root->data.n > newE.n)

root->left = addNewElement(root->left, newE);

else

root->right = addNewElement(root->right, newE);

}

return (root);

}

//==================Отобразить дерево==================

void showTree(Timet \*root, int prefix) {

if (root){

prefix++;

showTree(root->right, prefix);

int i; for(i=0; i != prefix; i++) printf("\t");

printf("%i\n", root->data.n);

for(i=0; i != prefix; i++) printf("\t");

printf("%s\n", root->data.so);

for(i=0; i != prefix; i++) printf("\t");

printf("%s\n", root->data.sp);

for(i=0; i != prefix; i++) printf("\t");

printf("%s\n", root->data.vo);

for(i=0; i != prefix; i++) printf("\t");

printf("%s\n", root->data.vp);

for(i=0; i != prefix; i++) printf("\t");

printf("%i\n", root->data.price);

showTree(root->left, prefix);

}

}

//==============Подсчитать число вершин на n-ом уровне дерева===========

int kolvoVershin(Timet \*root, int n, int search){

int kol = 0;

if (!root) {

return 0;

}

else{

if (n == search) kol++;

search++;

kol = kol + kolvoVershin(root->right, n, search);

kol = kol + kolvoVershin(root->left, n, search);

}

return kol;

}

//==================Взять инфо об одном новом элементе дерева==================

train dataInput() {

train newE;

int n;

printf("Введите номер поезда:\n");

scanf("%i", &n);

while ((getchar()) != '\n');

newE.n = n;

char so[21];

printf("Введите станцию отправления:\n");

gets(so);

strcpy(newE.so, so);

char sp[21];

printf("Введите станцию прибытся:\n");

gets(sp);

strcpy(newE.sp, sp);

char vo[6];

printf("Введите время отправления:\n");

gets(vo);

strcpy(newE.vo, vo);

char vp[6];

printf("Введите время прибытия:\n");

gets(vp);

strcpy(newE.vp, vp);

int Price;

printf("Введите стоимость билета:\n");

scanf("%i", &Price);

newE.price = Price;

return (newE);

}

//==================Cохранение дерева==================

void saveTree(FILE \*fout, Timet \*temp){

if (temp){

fprintf(fout, "%i\n", temp->data.n);

fprintf(fout, "%s\n", temp->data.so);

fprintf(fout, "%s\n", temp->data.sp);

fprintf(fout, "%s\n", temp->data.vo);

fprintf(fout, "%s\n", temp->data.vp);

fprintf(fout, "%i\n", temp->data.price);

saveTree(fout, temp->right);

saveTree(fout, temp->left);

}

printf("Данные успешно сохранены\n\n");

}

//==================Конец работы программы==================

void goodbye()

{

printf("================================= \n");

printf(" Программа окончила свою работу \n");

printf("================================= \n");

}

8.4.3 Выполнена отладка программы.

Результаты тестирования отображены на рисунках 8.3- 8.7

.

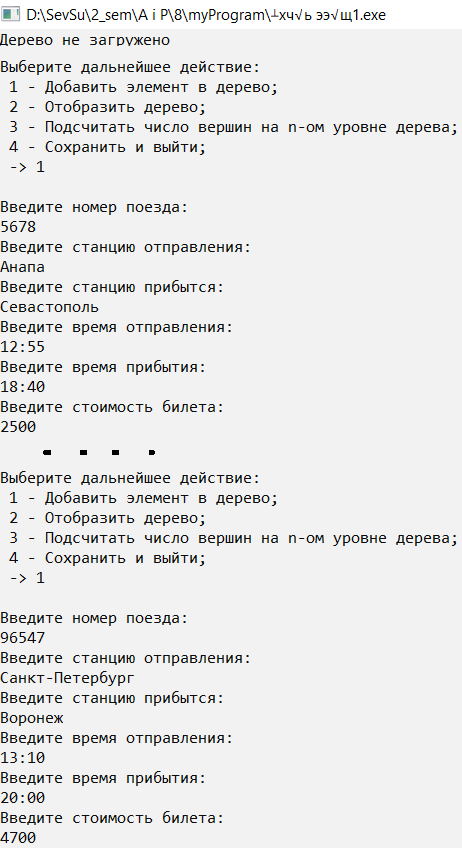


Рисунок 8.3 – Меню и добавление элементов структуры

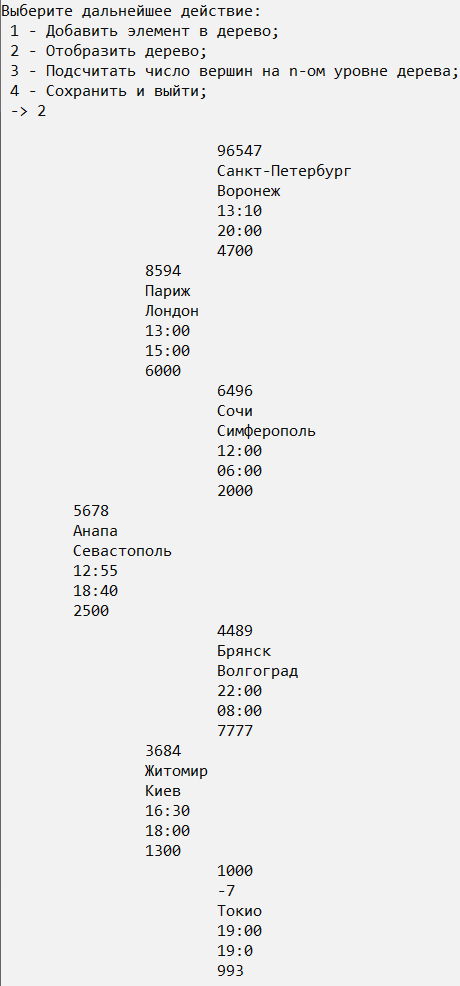


Рисунок 8.4 – Вывод элементов дерева

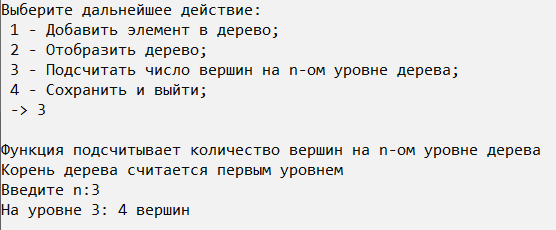


Рисунок 8.5 – Подсчитать количество вершин на n-ом уровне дерева

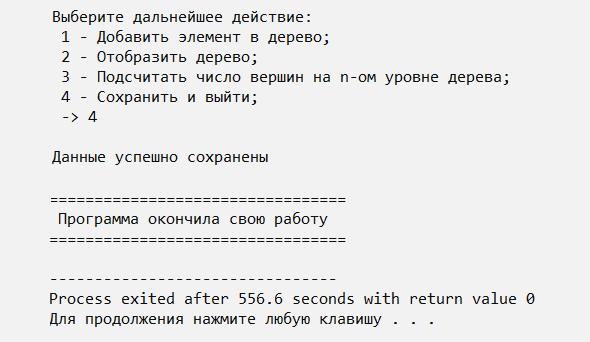


Рисунок 8.6 – Выход и сохранение

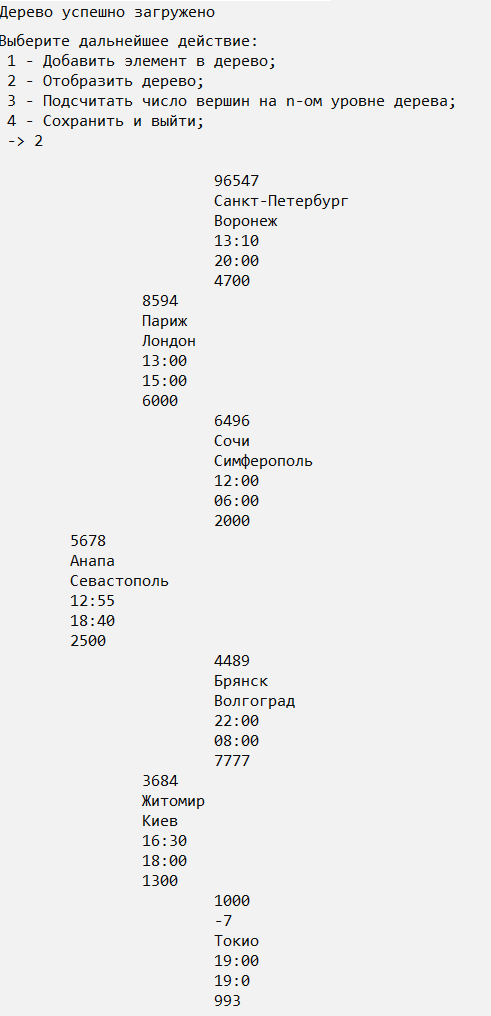


Рисунок 8.7 – Загрузка программы с загрузкой сохранённых данных

Результаты тестирования полностью соответствуют ожиданиям.

**Выводы**

В ходе выполнения данной лабораторной работы были получены и закреплены навыки разработки и отладки программ, использующих динамическую память. Были изучены способы организации деревьев на языке Си, а также способы удаления элементов дерева, добавления элементов в дерево и просмотр дерева. Исследованы способы сохранения и загрузки элементов дерева с помощью текстовых файлов. Полученные во время разработки навыки помогут разрабатывать более сложные программы со структурами данных и файлами, более эффективные по времени выполнения алгоритмы.