Министерство образования и науки Российской Федерации

Севастопольский Государственный Университет

Кафедра ИС

ОТЧЕТ

по лабораторной работе №8

ПРОГРАММИРОВАНИЕ НЕЛИНЕЙНЫХ

СТРУКТУР ДАННЫХ НА ЯЗЫКЕ C/C++

Выполнил: ст. гр. ИТб-11

Куркчи А.Э.

Проверил:

Забаштанский А.К.

Севастополь

2015

# 1. Цель

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++.

2. Постановка задач

Представить приведенную в предыдущей работе таблицу в виде бинарного дерева. Написать функции создания и обхода дерева, а также одну из функций, приведенных ниже. Значения полей и количество записей в таблице студент выбирает самостоятельно. Программа должна сохранять дерево в файле и создавать его заново при её повторном запуске.

Вариант №2

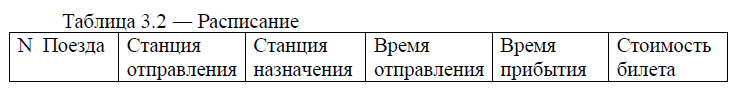


Таблица 3.2. Функцию, которая присваивает параметру Е элемент из самого левого листа непустого дерева Т.

3. Структурная схема

На рисунке 1 изображена схема алгоритма работы программы.

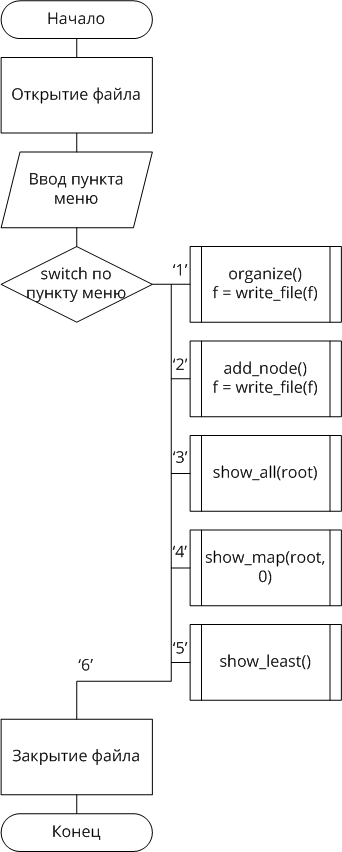
****

Рисунок 1 – Структурная схема

На рисунке 2 изображена схема алгоритма работы подпрограммы add\_node.

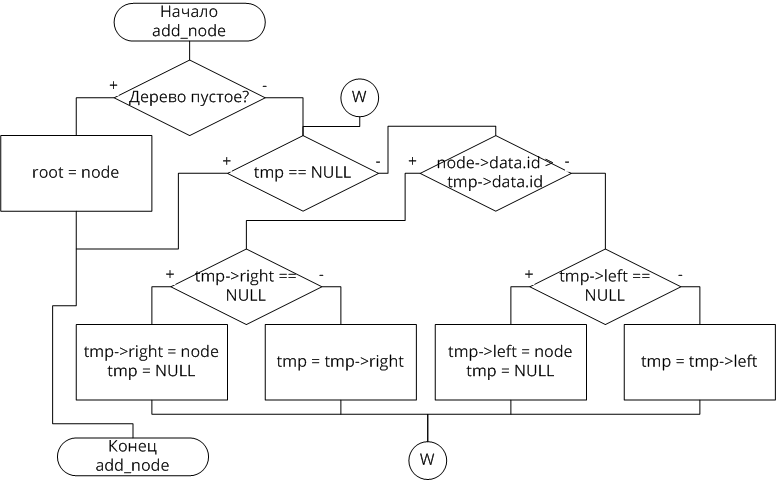
****

Рисунок 2 – Структурная схема add\_node

На рисунке 3 изображена схема алгоритма работы подпрограммы read\_file.

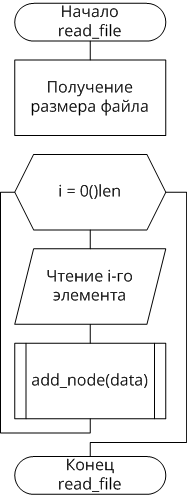
****

Рисунок 3 – Структурная схема read\_file

На рисунке 4 изображена схема алгоритма работы подпрограммы input.

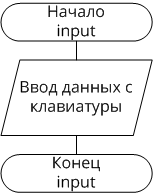
****

Рисунок 4 – Структурная схема input

На рисунке 5 изображена схема алгоритма работы подпрограммы organize.

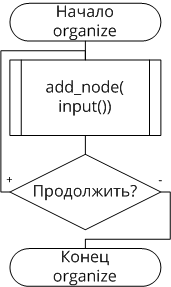
****

Рисунок 5 – Структурная схема organize

На рисунке 6 изображена схема алгоритма работы подпрограммы show\_all.

****

Рисунок 6 – Структурная схема show\_all

На рисунке 7 изображена схема алгоритма работы подпрограммы write\_data.

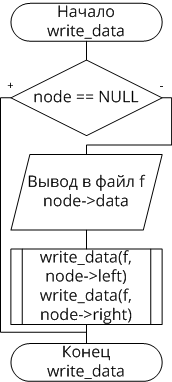
****

Рисунок 7 – Структурная схема write\_data

На рисунке 8 изображена схема алгоритма работы подпрограммы write\_file.

****

Рисунок 8 – Структурная схема write\_file

На рисунке 9 изображена схема алгоритма работы подпрограммы show\_map.

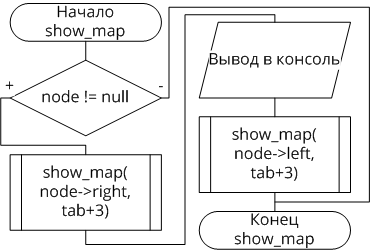
****

Рисунок 9 – Структурная схема show\_map

На рисунке 10 изображена схема алгоритма работы подпрограммы show\_least.

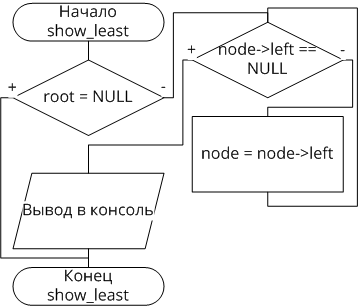
****

Рисунок 10 – Структурная схема show\_least

4. Текст программы

#include <stdio.h>

#include <iostream>

#include <iomanip>

#define STR\_SIZE 12

#define TIME\_SIZE 11

**using** **namespace** std**;**

struct data\_t **{**

int id**;**

char from**[**STR\_SIZE**];**

char to**[**STR\_SIZE**];**

char dep**[**TIME\_SIZE**];**

char ar**[**TIME\_SIZE**];**

float price**;**

**};**

struct node\_t **{**

data\_t data**;**

node\_t **\***left**,\***right**;**

**};**

const int size\_d **=** **sizeof(**data\_t**);**

node\_t **\***root**;**

// Добавить узел

void add\_node**(**data\_t data**)** **{**

node\_t **\***node **=** **new** node\_t**;**

node**->**data **=** data**;**

node**->**left **=** node**->**right **=** **NULL;**

**if(**root **==** **NULL)** **{**

root **=** node**;**

**}** **else** **{**

node\_t **\***tmp **=** root**;**

**while(**tmp **!=** **NULL)** **{**

**if(**node**->**data**.**id **>** tmp**->**data**.**id**)** **{**

**if(**tmp**->**right **==** **NULL)** **{**

tmp**->**right **=** node**;**

tmp **=** **NULL;**

**}** **else** **{**

tmp **=** tmp**->**right**;**

**}**

**}** **else** **{**

**if(**tmp**->**left **==** **NULL)** **{**

tmp**->**left **=** node**;**

tmp **=** **NULL;**

**}** **else** **{**

tmp **=** tmp**->**left**;**

**}**

**}**

**}**

**}**

**}**

// Чтение файла

void read\_file**(**FILE **\***f**)** **{**

fseek**(**f**,**0**,**SEEK\_END**);**

long len **=** ftell**(**f**)/**size\_d**;**

rewind**(**f**);**

root **=** **NULL;**

data\_t data**;**

**for(**int i**=**0**;**i**<**len**;**i**++)** **{**

fread**(&**data**,**size\_d**,**1**,**f**);**

add\_node**(**data**);**

**}**

**}**

// Запись узла в файл

void write\_data**(**FILE **\***f**,**node\_t **\***node**)** **{**

**if(**node **==** **NULL)** **{**

**return;**

**}**

fwrite**(&**node**->**data**,**size\_d**,**1**,**f**);**

write\_data**(**f**,**node**->**left**);**

write\_data**(**f**,**node**->**right**);**

**}**

// Запись файла

FILE**\*** write\_file**(**FILE **\***f**)** **{**

fclose**(**f**);**

f **=** fopen**(**"data.bin"**,**"w+b"**);**

write\_data**(**f**,**root**);**

**return** f**;**

**}**

// Ввод данных с клавиатуры

data\_t input**()** **{**

data\_t data**;**

cout **<<** "Train number:" **<<** endl**;**

cin **>>** data**.**id**;**

cin**.**get**();**

cout **<<** "Departure point:" **<<** endl**;**

cin**.**getline**(**data**.**from**,**STR\_SIZE**);**

cout **<<** "Arrival point:" **<<** endl**;**

cin**.**getline**(**data**.**to**,**STR\_SIZE**);**

cout **<<** "Departure time:" **<<** endl**;**

cin**.**getline**(**data**.**dep**,**TIME\_SIZE**);**

cout **<<** "Arrival time:" **<<** endl**;**

cin**.**getline**(**data**.**ar**,**TIME\_SIZE**);**

cout **<<** "Ticket price:" **<<** endl**;**

cin **>>** data**.**price**;**

cin**.**get**();**

**return** data**;**

**}**

// Организация очереди

void organize**()** **{**

char c**;**

**do** **{**

add\_node**(**input**());**

cout **<<** "Continue oranization? (y/n): "**;**

cin **>>** c**;**

cin**.**get**();**

**}** **while(**c **==** 'y'**);**

**}**

// Печать дерева

void show\_all**(**node\_t **\***node**)** **{**

**if(**root **==** **NULL)** **{**

cout **<<** "Tree is empty" **<<** endl**;**

cout **<<** "Press enter to continue..." **<<** endl**;**

cin**.**get**();**

**return;**

**}**

**if(**node **==** **NULL)** **{**

**return;**

**}**

**if(**node **==** root**)** **{**

cout **<<** "+---+------------+------------+-----------+-----------+---------+" **<<** endl**;**

cout **<<** "| # | From | To | Departure | Arrival | Price |" **<<** endl**;**

**}**

cout**.**precision**(**2**);**

cout **<<** "+---+------------+------------+-----------+-----------+---------+" **<<** endl**;**

cout **<<** "|" **<<** setw**(**3**)** **<<** node**->**data**.**id **<<** "|" **<<** setw**(**12**)** **<<** node**->**data**.**from **<<** "|" **<<** setw**(**12**)** **<<** node**->**data**.**to **<<** "|" **<<** setw**(**11**)** **<<** node**->**data**.**dep **<<** "|" **<<** setw**(**11**)** **<<** node**->**data**.**ar **<<** "|" **<<** setw**(**9**)** **<<** node**->**data**.**price **<<** "|" **<<** endl**;**

show\_all**(**node**->**left**);**

show\_all**(**node**->**right**);**

**if(**node **==** root**)** **{**

cout **<<** "+---+------------+------------+-----------+-----------+---------+" **<<** endl**;**

cout **<<** "Press enter to continue..." **<<** endl**;**

cin**.**get**();**

**}**

**}**

// Печать структуры дерева

void show\_map**(**node\_t **\***node**,**int tab**)** **{**

**if(**tab **==** 0**)** **{**

**if(**node **==** **NULL)** **{**

cout **<<** "Tree is empty" **<<** endl**;**

**}** **else** **{**

cout **<<** "Tree structure:" **<<** endl**;**

**}**

**}**

**if(**node **!=** **NULL)** **{**

show\_map**(**node**->**right**,**tab**+**3**);**

cout **<<** setw**(**tab**)** **<<** " " **<<** node**->**data**.**id **<<** endl**;**

show\_map**(**node**->**left**,**tab**+**3**);**

**}**

**if(**tab **==** 0**)** **{**

cout **<<** "Press enter to continue..." **<<** endl**;**

cin**.**get**();**

**}**

**}**

// Печать дерева

void show\_least**()** **{**

**if(**root **==** **NULL)** **{**

cout **<<** "Tree is empty" **<<** endl**;**

cout **<<** "Press enter to continue..." **<<** endl**;**

cin**.**get**();**

**return;**

**}**

node\_t **\***node **=** root**;**

**while(**node**->**left **!=** **NULL)** **{**

node **=** node**->**left**;**

**}**

cout **<<** "+---+------------+------------+-----------+-----------+---------+" **<<** endl**;**

cout **<<** "| # | From | To | Departure | Arrival | Price |" **<<** endl**;**

cout **<<** "+---+------------+------------+-----------+-----------+---------+" **<<** endl**;**

cout**.**precision**(**2**);**

cout **<<** "|" **<<** setw**(**3**)** **<<** node**->**data**.**id **<<** "|" **<<** setw**(**12**)** **<<** node**->**data**.**from **<<** "|" **<<** setw**(**12**)** **<<** node**->**data**.**to **<<** "|" **<<** setw**(**11**)** **<<** node**->**data**.**dep **<<** "|" **<<** setw**(**11**)** **<<** node**->**data**.**ar **<<** "|" **<<** setw**(**9**)** **<<** node**->**data**.**price **<<** "|" **<<** endl**;**

cout **<<** "+---+------------+------------+-----------+-----------+---------+" **<<** endl**;**

cout **<<** "Press enter to continue..." **<<** endl**;**

cin**.**get**();**

**}**

int main**(**int argc**,** char**\*\*** argv**)** **{**

FILE **\***f**;**

f **=** fopen**(**"data.bin"**,**"r+b"**);**

**if(!**f**)** **{**

f **=** fopen**(**"data.bin"**,**"w+b"**);**

**if(!**f**)** **{**

cout **<<** "Can`t open (create) file" **<<** endl**;**

**return** 1**;**

**}**

**}**

read\_file**(**f**);**

char c**;**

**while(**1**)** **{**

cout **<<** "1 - Organize" **<<** endl**;**

cout **<<** "2 - Add node" **<<** endl**;**

cout **<<** "3 - Show elements" **<<** endl**;**

cout **<<** "4 - Show structure" **<<** endl**;**

cout **<<** "5 - Show the least element" **<<** endl**;**

cout **<<** "6 - Exit" **<<** endl**;**

cout **<<** "Choose your destiny: "**;**

cin **>>** c**;**

cin**.**get**();**

**switch(**c**)** **{**

**case** '1'**:**

organize**();**

f **=** write\_file**(**f**);**

**break;**

**case** '2'**:**

add\_node**(**input**());**

f **=** write\_file**(**f**);**

**break;**

**case** '3'**:**

show\_all**(**root**);**

**break;**

**case** '4'**:**

show\_map**(**root**,**0**);**

**break;**

**case** '5'**:**

show\_least**();**

**break;**

**case** '6'**:**

fclose**(**f**);**

**return** 0**;**

**}**

**}**

fclose**(**f**);**

**return** 0**;**

**}**

5. Тестовые примеры

На рисунке 11 представлено меню программы

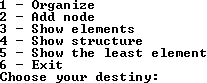


Рисунок 11 – Меню программы

На рисунке 12 представлен вывод пункта меню 3

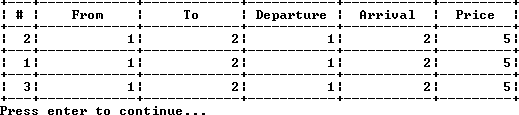


Рисунок 12 – Вывод всех элементов дерева

На рисунке 13 представлен вывод пункта меню 4

http://pastexen.com/i/JEbcFLKpIO.png

Рисунок 13 – Вывод структуры дерева

Вывод

В ходе лабораторной работы были изучены нелинейные структуры данных. Приобретены навыки разработки и отладки программ, использующих древовидные структуры. Исследованы особенности работы с поисковыми бинарными деревьями на языке C/C++.