Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №4

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Бинарное дерево поиска»

**Выполнили студенты группы 21вв1.1:**

Кузнецов А.С.

Гришин С.В.

**Приняли**

Юрова О.В.

Акифьев И.В.

Пенза 2022

**Методические указания.**

1. Реализовать алгоритм поиска вводимого с клавиатуры значения в уже созданном дереве.
2. Реализовать функцию подсчёта числа вхождений заданного элемента в дерево.
3. \* Изменить функцию добавления элементов для исключения добавления одинаковых символов.
4. \* Оценить сложность процедуры поиска по значению в бинарном дереве.

**Описание метода решения задачи:**

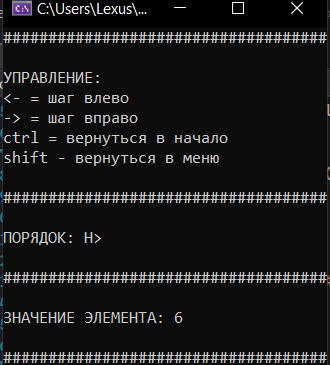
1)Реализовать структуры бинарного древа .

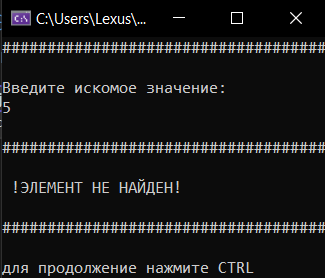
2) Написать функции для работы с ним .

3)Добавить функцию изменения добавление элементов для исключению одинаковых символов.

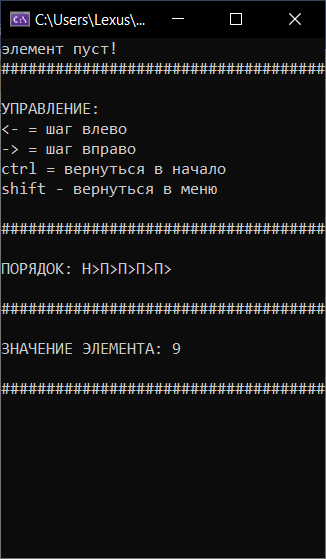
**Результаты работы программы:**

1)Реализовали алгоритм поиска вводимого с клавиатуры значения в созданном дереве.

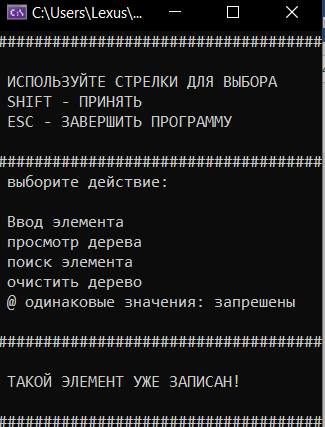




2) Реализована функция подсчёта числа вхождений заданного элемента в дерево.



3) Изменили функцию добавления элементов для исключения добавления одинаковых символов.



4) Оценить сложность процедуры поиска по значению в бинарном дереве.

Сложность алгоритма поиска составляет O(log(N))

**Вывод:** в ходе выполнения лабораторной работы, мы научились работать с бинарным деревом : выводить и вводить элементы ,выполнять поиск в нём .

**Листинг:**

#include <iostream>

#include <Windows.h>

#include <stdlib.h>

void hidecursore()

{

HANDLE consolehandle = GetStdHandle(STD\_OUTPUT\_HANDLE);

CONSOLE\_CURSOR\_INFO info;

info.dwSize = 100;

info.bVisible = FALSE;

SetConsoleCursorInfo(consolehandle, &info);

}

typedef struct bts

{

int zn;

struct bts\* left;

struct bts\* right;

};

struct bts\* init(int zn)

{

struct bts\* bf;

bf = (struct bts\*)malloc(sizeof(struct bts));

bf->zn = zn;

bf->left = NULL;

bf->right = NULL;

return(bf);

}

void add(bts\* bt, int zn)

{

struct bts\* bf = bt;

bool chek = false;

while (chek == false)

{

if (bt->zn < zn)

{

if (bt->right != NULL)

{

bt = bt->right;

}

else

{

struct bts\* newb;

newb = (struct bts\*)malloc(sizeof(struct bts));

newb->zn = zn;

newb->left = NULL;

newb->right = NULL;

bt->right = newb;

chek = true;

}

}

else

{

if (bt->left != NULL)

{

bt = bt->left;

}

else

{

struct bts\* newb;

newb = (struct bts\*)malloc(sizeof(struct bts));

newb->zn = zn;

newb->left = NULL;

newb->right = NULL;

bt->left = newb;

chek = true;

}

}

}

}

void ched(bts\* bt)

{

system("cls");

bool cic = false;

char mov[100];

int mc = 1;

for (int i = 0; i < 100; i++) { mov[i] = '\0'; }

mov[0] = 'Н';

struct bts\* bf = bt;

lb1:

printf("####################################\n\n");

printf("УПРАВЛЕНИЕ:\n<- = шаг влево\n-> = шаг вправо\nctrl = вернуться в начало\nshift - вернуться в меню\n");

printf("\n####################################\n");

printf("\nПОРЯДОК: ");

int i = 0;

while (mov[i] != '\0')

{

printf("%c>", mov[i]);

i++;

}

printf("\n\n####################################\n\n");

printf("ЗНАЧЕНИЕ ЭЛЕМЕНТА: %d", bf->zn);

printf("\n\n####################################\n");

cic = false;

endf:

while (cic == false)

{

printf("");

Sleep(100);

if (GetKeyState(37) < 0)

{

system("cls");

if (bf->left != NULL)

{

bf = bf->left;

mov[mc] = 'Л';

mc++;

system("cls");

goto lb1;

}

else

{

printf("элемент пуст!\n");

goto lb1;

}

}

if (GetKeyState(39) < 0)

{

system("cls");

if (bf->right != NULL)

{

bf = bf->right;

mov[mc] = 'П';

mc++;

system("cls");

goto lb1;

}

else

{

printf("элемент пуст!\n");

goto lb1;

}

}

if (GetKeyState(17) < 0)

{

bf = bt;

for (int i = 0; i < 100; i++) { mov[i] = '\0'; }

mov[0] = 'Н';

mc = 1;

system("cls");

goto lb1;

}

if (GetKeyState(16) < 0)

{

bf = bt;

for (int i = 0; i < 100; i++) { mov[i] = '\0'; }

mov[0] = 'Н';

mc = 1;

system("cls");

cic = true;

goto endf;

}

}

}

int search(bts\* bt, int zn)

{

bts\* bf = bt;

while (bf != 0)

{

if (bf->zn == zn) { return(1); }

else

{

if (zn > bf->zn) { bf = bf->right; }

else { bf = bf->left; }

}

}

if (bf == NULL) { return(0); }

}

void sermod(bts\* bt, int zn)

{

bts\* bf = bt;

printf("\n####################################\n");

char mov[100];

for (int i = 0; i < 100; i++) { mov[i] = '\0'; }

mov[0] = 'Н';

int i = 1;

while (bf != 0)

{

if (bf->zn == zn)

{

printf("\n !ЭЛЕМЕНТ НАЙДЕН!\n");

printf("\n####################################\n\n");

printf("ПУТЬ: ");

int ns = 0;

while (mov[ns + 1] != '\0') { printf("%c>", mov[ns]); ns++; }

printf("%c", mov[ns]);

printf("\n\n####################################");

goto end;

}

else

{

if (zn > bf->zn) { bf = bf->right; mov[i] = 'П'; i++; }

else { bf = bf->left; mov[i] = 'Л'; i++; }

}

}

if (bf == NULL) {

printf("\n !ЭЛЕМЕНТ НЕ НАЙДЕН!\n");

printf("\n####################################\n\n");

}

end:

printf("для продолжение нажмите CTRL");

bool chec = false;

while (chec == false)

{

if (GetKeyState(17) < 0) { chec = true; }

}

}

int main()

{

hidecursore();

system("mode con cols=36 lines=16");

system("chcp 1251");

system("cls");

int mod = 1, zn = 0;

int mesg = 0;

bool chec = false, clone = true;

struct bts\* bt = NULL;

lm:

printf("####################################\n");

printf("\n ИСПОЛЬЗУЙТЕ СТРЕЛКИ ДЛЯ ВЫБОРА\n");

printf(" SHIFT - ПРИНЯТЬ\n");

printf(" ESC - ЗАВЕРШИТЬ ПРОГРАММУ\n");

printf("\n####################################\n");

printf(" выборите действие: \n\n");

if (mod == 1) { printf(" @ Ввод элемента\n просмотр дерева\n поиск элемента\n очистить дерево\n одинаковые значения:"); }

if (mod == 2) { printf(" Ввод элемента\n @ просмотр дерева\n поиск элемента\n очистить дерево\n одинаковые значения:"); }

if (mod == 3) { printf(" Ввод элемента\n просмотр дерева\n @ поиск элемента\n очистить дерево\n одинаковые значения:"); }

if (mod == 4) { printf(" Ввод элемента\n просмотр дерева\n поиск элемента\n @ очистить дерево\n одинаковые значения:"); }

if (mod == 5) { printf(" Ввод элемента\n просмотр дерева\n поиск элемента\n очистить дерево\n @ одинаковые значения:"); }

if ((clone == true)) { printf(" разрешены\n"); }

else { printf(" запрешены\n"); }

printf("\n####################################");

if (mesg == 1)

{

printf("\n\n");

printf(" ЭЛЕМЕНТ УСПЕШНО ЗАПИСАН!\n");

printf("\n####################################");

}

if (mesg == 2)

{

printf("\n\n");

printf(" ТАКОЙ ЭЛЕМЕНТ УЖЕ ЗАПИСАН!\n");

printf("\n####################################");

}

if (mesg == 3)

{

printf("\n\n");

printf(" ДЕРЕВО ПУСТОЕ!\n");

printf("\n####################################");

}

if (mesg == 4)

{

printf("\n\n");

printf(" ДЕРЕВО ОЧИЩЕННО!\n");

printf("\n####################################");

}

while (chec == false)

{

Sleep(60);

if ((GetKeyState(38) < 0) and (mod > 1)) { mod--; system("cls"); goto lm; }

if ((GetKeyState(40) < 0) and (mod < 5)) { mod++; system("cls"); goto lm; }

if (GetKeyState(16) < 0) { goto next; }

if (GetKeyState(27) < 0) { exit(EXIT\_SUCCESS); }

}

next:

if (mod == 2)

{

if (bt == NULL) { mesg = 3; system("cls"); system("mode con cols=36 lines=20"); goto lm; }

else

{

system("mode con cols=36 lines=26");

ched(bt);

system("mode con cols=36 lines=16");

mesg = 0;

goto lm;

}

}

if (mod == 1)

{

system("cls");

printf("Введите значение:\n");

scanf\_s("%d", &zn);

if (clone == false)

{

int prov;

prov = search(bt, zn);

if (prov == 0) { goto ok; }

else

{

system("mode con cols=36 lines=20");

mesg = 2;

goto lm;

}

}

ok:

if (bt == NULL) { bt = init(zn); }

else { add(bt, zn); }

mesg = 1;

system("cls");

system("mode con cols=36 lines=20");

goto lm;

}

if (mod == 3)

{

system("mode con cols=36 lines=26");

system("cls");

printf("####################################\n\n");

printf("Введите искомое значение: \n");

scanf\_s("%d", &zn);

sermod(bt, zn);

system("cls");

system("mode con cols=36 lines=16");

mesg = 0;

goto lm;

}

if (mod == 4)

{

bt = NULL;

mesg = 4;

system("cls");

system("mode con cols=36 lines=20");

goto lm;

}

if (mod == 5) { if (clone == true) { clone = false; } else { clone = true; }; system("cls"); goto lm; }

}