ГУАП КАФЕДРА № 34

ОТЧЕТ

ЗАЩИЩЕН С ОЦЕНКОЙ ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Старший преподаватель |  |  |  | К.А. Жиданов |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2 |
| Язык Си |
| по курсу: ЯЗЫКИ ПРОГРАММИРОВАНИЯ |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |
| --- | --- | --- |
| СТУДЕНТ ГР. № | 3145 17.06.2022 | Щербак. А.А |
|  | подпись, дата | инициалы, фамилия |

Санкт-Петербург 2021

**Задание:** Реализовать АТД (абстрактный тип данных) в виде пользовательского типа данных и

набора функций, реализующих заданные операции. Бинарное дерево поиска (добавление, поиск)

**Ход работы:**

Бинарное дерево поиска состоит из узлов. Каждый узел содержит в себе указатели на левое и правое поддерево, указатель на родителя и ключ. Узлы представляются в качестве структуры:

typedef struct node {

int key;

struct node\* left;

struct node\* right;

} node\_tr;

Используем typedef для создания нового типа, чтобы в дальнейшем не писать слово struct.

int key — ключ, может быть любого типа.

struct node \*left — указатель на левое поддерево.

struct node \*right — указатель на правое поддерево.

node\_tr — название структуры.

Если ветви нет, то вместо ветви указываем на NULL.

Инициализируем дерево отдельной функцией. Проще говоря, создаем корень бинарного дерева поиска.

Добавляем узлы в дерево: выделяем память под узел и присваиваем значение ключу. С помощью функции while присваиваем значение указателя на родителя значение дочернего указателя. Присваиваем указателю на левое и правое поддерево значение NULL и вставляем узел в дерево.

Найдем нужный узел в дереве. Для этого руководствуемся правилом: слева расположены элементы с меньшим значением ключа, справа — с большим. Однако, если дерево окажется пустым или ключ корня равен искомому ключу, то возвращается указатель на корень.

Для удобства находим ключ с максимальным и минимальным значениями.

Выбираем элементы в дереве:

int test(char\* name, char type) {

node\_tr\* x = NULL;

if (NULL != tree\_s(x, 2)) {

return 1;

}

x = tree\_c(2);

if (NULL == tree\_s(x, 2)) {

return 2;

}

tree\_a(x, 20);

if (NULL == tree\_s(x, 20)) {

return 3;

}

if (type == "w") {

FILE\* fp = fopen(name, type);

if (x == NULL) {

return 4;

}

serialize(x, fp);

fclose(fp);

}

if (type == 'r') {

node\_tr\* y = NULL;

FILE\* fn = fopen(name, type);

if (fn == NULL) {

return 5;

}

y = deserialize(y, fn);

if (NULL == y) {

return 6;

}

}

return 0;

}

Если все работает корректно (вернется 0), то возвращается функция test

**Вывод:** мы изучили методы реализации АТД в виде пользовательского типа данных и набора функций, реализующих заданные операции. Научились добавлять значения и искать в бинарном дереве