Санкт-Петербургский государственный политехнический университет

Кафедра “Прикладная математика”

**Отчет по лабораторной работе 5**

**“Алгоритмы и структуры данных”**

Студент группы № 5030102/20001

ФИО: Королева Дарья Сергеевна

Выполнила (дата) 15.12.2023

Оглавление

[**Постановка задачи 3**](#_Toc153420310)

[**Описание алгоритма 3**](#_Toc153420311)

[**Текст программы 5**](#_Toc153420312)

[**Описание тестирования 10**](#_Toc153420313)

# **Постановка задачи**

Вариант 9:

Для дерева двоичного поиска с дополнительной информацией реализовать процедуру, которая находит K-й наименьший элемент и, если он оказывается четным, печатает элементы с порядковыми номерами меньшими K.

# **Описание алгоритма**

Алгоритм создает дерево, путем выделения памяти под структуру,  
являющуюся узлом дерева и содержащую следующую информацию:

* Значение
* Порядковый номер
* Адрес на левый дочерний элемент
* Адрес на родительский элемент
* Адрес на правый дочерний элемент

и заполняя ее. Каждый новый узел, т е подобная структура, вставляется на  
соответствующую ему позицию в дереве в соответствии с его значением. Для  
нахождения узла с нужным порядковым номером проходится все дерево и сравниваются значения, при нахождении результатом является указатель на нужный узел. В конце при четности найденного числа выводятся на экран, порядковые номера которых меньше найденного. В конце дерево удаляется, рекурсивно удаляя все элементы.

# **Текст программы**

#pragma warning(disable:4996)

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

typedef struct {

int num;

int order;

struct tree\_t\* left;

struct tree\_t\* parent;

struct tree\_t\* right;

int subtree\_wigth;

} tree\_t;

typedef enum {

NONE = 0,

LEFT = 1,

RIGHT = 2

}variant\_t;

void orderUpdate(tree\_t\* tree, int\* order) {

if (tree) {

if (tree->left)

orderUpdate(tree->left, order);

tree->order = \*order;

(\*order)++;

if (tree->right)

orderUpdate(tree->right, order);

}

}

void addition(tree\_t\*\* tree, int num) {

tree\_t\* temp = (\*tree), \* newnode, \* ptr = (\*tree);

int order = 1;

newnode = (tree\_t\*)malloc(sizeof(tree\_t));

if (!newnode)

return;

newnode->left = NULL;

newnode->right = NULL;

newnode->parent = NULL;

newnode->num = num;

if (!(\*tree)) {

(\*tree) = newnode;

orderUpdate((\*tree), &order);

return;

}

while (ptr) {

if (num < ptr->num) {

if (ptr->left)

ptr = ptr->left;

else

break;

}

else if (num > ptr->num) {

if (ptr->right)

ptr = ptr->right;

else

break;

}

else {

free(newnode);

return;

}

}

if (num < ptr->num)

ptr->left = newnode;

else

ptr->right = newnode;

newnode->parent = ptr;

orderUpdate((\*tree), &order);

}

void destroyTree(tree\_t\* tree) {

if (tree) {

destroyTree(tree->left);

destroyTree(tree->right);

free(tree);

}

}

tree\_t\* KlowerFind(tree\_t\* tree, int k) {

if (tree) {

if (tree->order == k)

return tree;

if (KlowerFind(tree->left, k))

return tree->left;

if (KlowerFind(tree->right, k))

return tree->right;

}

return NULL;

}

void PrintTree(tree\_t\* tree, int k) {

if (tree) {

PrintTree(tree->left, k);

if (tree->order < k) {

printf("%d ", tree->num);

}

PrintTree(tree->right, k);

}

}

int main(void) {

const int k = 3;

tree\_t\* tree = NULL, \* find = NULL;

addition(&tree, 5);

addition(&tree, 2);

addition(&tree, -1);

addition(&tree, -6);

addition(&tree, 10);

addition(&tree, 3);

find = KlowerFind(tree, k);

printf("The smallest number: %d\n", find->num);

if (find && (find->num) % 2 == 0)

PrintTree(tree, k);

else printf("odd number");

destroyTree(tree);

return 0;

}

# **Описание тестирования**

Для тестирования данной программы производился ее неоднократный запуск с вводом конкретных данных, при котором проверялось:

1. Стабильность работы программы при одинаковых входных данных;
2. Корректное выполнение всех заявленных процедур;
3. Корректное завершение программы;

Приведем пример тестирования на скриншотах, представленных ниже:

addition(&tree, 5);

addition(&tree, 2);

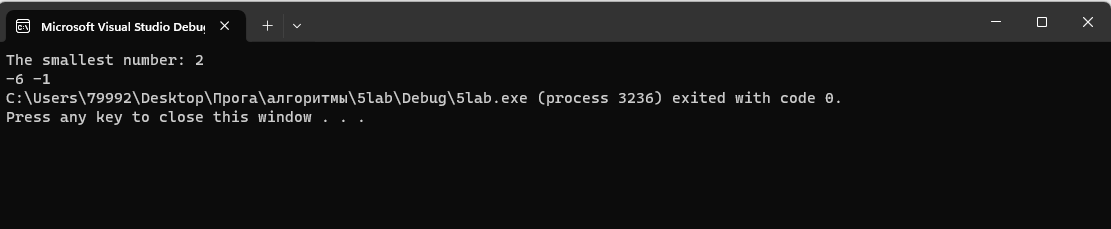
addition(&tree, -1);

addition(&tree, -6);

addition(&tree, 10);

addition(&tree, 3);

const int k = 3;



const int k = 6;

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

const int k = 5;

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

При проведении тестирования такого рода никаких проблем обнаружено не было, что позволяет судить о корректности работы программы в целом.