**Практична робота № 15-16**

*Виконав студент 35 групи*

*Рижій Ілля*

**Завдання №1.** Реалізувати *приклад\_1.*

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#include <stdlib.h>

#define MAXWORD 100

struct tnode{ // вузол дерева

char \*word; // покажчик на слово

int count; // число входжень

struct tnode \*left; // лівий нащадок

struct tnode \*right; // правий нащадок

};

// Функція додавання вузла до дерева

struct tnode \*addtree(struct tnode \*p, char \*w){

int cond;

if(p == NULL){

p = (struct tnode \*)malloc(sizeof(struct tnode));

p->word = strdup(w);

p->count = 1;

p->left = p->right = NULL;

}

else if((cond = strcmp(w, p->word)) == 0)

p->count++;

else if(cond < 0)

p->left = addtree(p->left, w);

else

p->right = addtree(p->right, w);

return p;

}

// Функція видалення піддерева

void freemem(tnode \*tree){

if(tree != NULL){

freemem(tree->left);

freemem(tree->right);

free(tree);

}

}

// Функція виведення дерева

void treeprint(struct tnode \*p){

if(p != NULL){

treeprint(p->left);

printf("%d %s\n", p->count, p->word);

treeprint(p->right);

}

}

int main(){

struct tnode \*root;

char word[MAXWORD];

root = NULL;

printf("Enter the string. Exit condition - character input '0' \n");

do{

scanf("%s", word);

if(isalpha(word[0]))

root = addtree(root, word);

}while(word[0] != '0'); // умова виходу

treeprint(root);

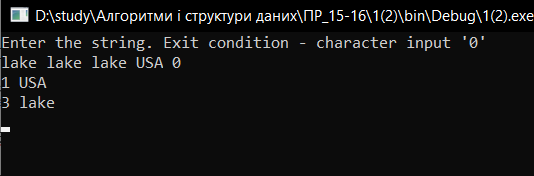
freemem(root);

getchar();

getchar();

return 0;

}



**Завдання №2.** Написати програму, яка визначає число входжень однакових цифр у число, введене з клавіатури.

#include <bits/stdc++.h>

using namespace std;

// tree nod

struct Node

{

int data;

int key ,count;

Node \*left, \*right;

};

// returns a new tree Node

Node\* newNode(int data)

{

struct Node \*temp = (struct Node \*)malloc(sizeof(struct Node));

temp->key = data;

temp->left = temp->right = NULL;

temp->count = 1;

return temp;

}

// A function to create binary search tree.

Node\* Tree(Node\* temp, int key )

{

// If the tree is empty, return a new node

if (temp == NULL)

return newNode(key);

if (key == temp->key)

{

(temp->count)++;

return temp;

}

// Otherwise, recur down the tree

if (key < temp->key)

temp->left = Tree(temp->left, key);

else

temp->right = Tree(temp->right, key);

//return the (unchanged) node pointer

return temp;

}

//function to display all the element present in the binary search tree

void displayTree(struct Node\* root)

{

if (root != NULL)

{

displayTree(root->left);

cout<<"Number "<<root->key<<" duplicated times: "<<root->count<<endl;

displayTree(root->right);

}

}

int main()

{

int arr[20],size;

Node \*root = new Node;

root = NULL;

cout<<"Enter the size of array : ";

cin>>size;

cout<<"Enter the elements in array : ";

for(int i = 0; i < size; i++)

{

cin>>arr[i];

}

// Construct the binary tree.

for(int i = 0; i < size; i++)

{

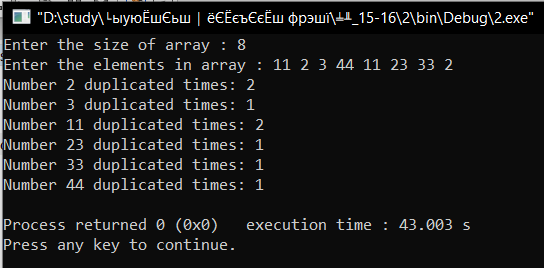
root = Tree(root, arr[i]);

}

displayTree(root);

return 0;

}



**Завдання №3**. Скласти програму для побудови і виведення на екран ідеально збалансованого дерева.

#include <bits/stdc++.h>

using namespace std;

struct node

{

int info; //Информационное поле

node \*l, \*r;//Левая и Правая часть дерева

};

node \*tree = NULL; //Объявляем переменную, тип которой структура Дерево

/\*ФУНКЦИЯ ЗАПИСИ ЭЛЕМЕНТА В БИНАРНОЕ ДЕРЕВО\*/

void push(int a, node \*\*t)

{

if ((\*t) == NULL) //Если дерева не существует

{

(\*t) = new node; //Выделяем память

(\*t)->info = a; //Кладем в выделенное место аргумент a

(\*t)->l = (\*t)->r = NULL; //Очищаем память для следующего роста

return;

}

//Дерево есть

if (a > (\*t)->info) push(a, &(\*t)->r); //Если аргумент а больше чем текущий элемент, кладем его вправо

else push(a, &(\*t)->l); //Иначе кладем его влево

}

/\*ФУНКЦИЯ ОТОБРАЖЕНИЯ ДЕРЕВА НА ЭКРАНЕ\*/

void print (node \*t)

{

if (t == NULL)

return; //Если дерево пустое, то отображать нечего, выходим

else //Иначе

{

print(t->l); //С помощью рекурсивного посещаем левое поддерево

cout<<t->info<<" "; //И показываем элемент

}

print(t->r); //С помощью рекурсии посещаем правое поддерево

}

int main ()

{

setlocale(LC\_ALL,"");

int n; //Количество элементов

int s; //Число, передаваемое в дерево

cout<<"Введите количество элементов: ";

cin>>n; //Вводим количество элементов

for (int i = 0; i < n; ++i)

{

cout<<"Введите число: ";

cin>>s; //Считываем элемент за элементом

push(s, &tree); //И каждый кладем в дерево

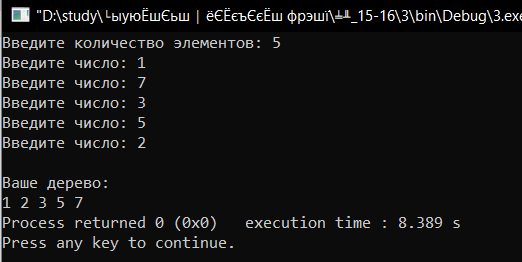
}

cout<<"\nВаше дерево: \n";

print(tree);

return 0;

}



**Завдання №4**. Скласти програму для побудови і виведення на екран дерева пошуку.

#include <iostream>

using namespace std;

struct node {

int key;

struct node \*left, \*right;

};

// Create a node

struct node \*newNode(int item) {

struct node \*temp = (struct node \*)malloc(sizeof(struct node));

temp->key = item;

temp->left = temp->right = NULL;

return temp;

}

// Insert a node

struct node \*insert(struct node \*node, int key) {

// Return a new node if the tree is empty

if (node == NULL) return newNode(key);

if (key < node->key)

node->left = insert(node->left, key);

else

node->right = insert(node->right, key);

return node;

}

void displayTree(struct node \*root) {

if (root != NULL) {

displayTree(root->left);

cout << root->key << " -> ";

displayTree(root->right);

}

}

int main() {

struct node \*root = NULL;

root = insert(root, 8);

root = insert(root, 3);

root = insert(root, 1);

root = insert(root, 6);

root = insert(root, 7);

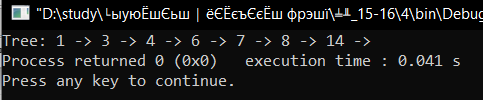
root = insert(root, 14);

root = insert(root, 4);

cout << "Tree: ";

displayTree(root);

}



**Висновок:** під час лабораторної роботи закріпили знання про бінарні дерева.