**SPRAWOZDANIE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Przedmiot** | Wprowadzenie do Informatyki | **Zadanie** | 6.1 |
| **Autor** | Andrii Godlevskyi | **Grupa** | WCY23KY1S1 |
| **Temat** | Struktury dynamiczne​ | | |

1.Treść zadania

Należy napisać program, który umożliwi:

Budowanie drzewa BST.

Program powinien umożliwić:

* Dodawanie węzła (wprowadzanie danych z klawiatury)
* Wyszukanie węzła (podanie wszystkich jego przodków od korzenia)
* Wypisanie węzłów drzewa w kolejności przechodzenia.

Metoda realizacji

Tworzymy strukturę drzewa i funkcje operujące na nim i w zależności od wyboru użytkownika wywołujemy potrzebną funkcje i wyprowadzamy wynik na ekran.

1. Założenia / ograniczenia dotyczące danych:

1. Dane wejściowe

Dane węzłów – wczytywane z klawiatury

1. Dane wyjściowe

Dane zapisanych węzłów - wyświetlone na ekranie *(z opisami).*

1. Realizacja
2. Algorytm

Изображение выглядит как текст, рукописный текст, зарисовка, рисунок

Автоматически созданное описание



1. Kod zródłowy

#include <stdio.h>

#include <stdlib.h>

// Deklaracja struktury

struct node {

int data;

struct node\* left;

struct node\* right;

struct node\* parent; // Dodane pole parent

};

// Funkcja tworzenia wezla

struct node\* createNode(int data) {

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

newNode->data = data;

newNode->left = NULL;

newNode->right = NULL;

newNode->parent = NULL; // Inicjalizacja pola parent

return newNode;

}

// Funkcja dodania wezla w odpowiedniej kolejnosci

struct node\* insertNode(struct node\* root, int data) {

if (root == NULL) {

return createNode(data);

}

if (data < root->data) {

root->left = insertNode(root->left, data);

root->left->parent = root; // Ustawienie rodzica dla lewego dziecka

} else if (data > root->data) {

root->right = insertNode(root->right, data);

root->right->parent = root; // Ustawienie rodzica dla prawego dziecka

}

return root;

}

// Funkcja znajdowania węzła

struct node\* searchNode(struct node\* root, int data) {

if (root == NULL || root->data == data) {

return root;

}

if (data < root->data) {

return searchNode(root->left, data);

} else {

return searchNode(root->right, data);

}

}

// Funkcja wypisywania przodkow wezla

void printAncestors(struct node\* node) {

while (node != NULL) {

printf("%d ", node->data);

node = node->parent;

}

printf("\n");

}

// Funkcja wypisania wezlow w odpowiedniej kolejnosci

void printInorder(struct node\* root) {

if (root != NULL) {

printInorder(root->left);

printf("%d ", root->data);

printInorder(root->right);

}

}

int main() {

// Deklaracja zmiennych

struct node\* root = NULL;

int choice, data;

// Menu wyboru

do {

printf("\n-------------menu-------------\n1. Dodaj wezel\n2. Znajdz wezel\n3. Wypisz kolejnosc przechodzenia \n4. Wyjscie\n");

printf("Wpisz swoj wybor: ");

scanf("%d", &choice);

switch(choice) {

case 1:

printf("Wpisz nowy wezel: ");

scanf("%d", &data);

root = insertNode(root, data);

break;

case 2:

printf("Wpisz wezel, ktory chcesz znalezc: ");

scanf("%d", &data);

struct node\* found = searchNode(root, data);

if (found != NULL) {

printf("Wezel znajduje się w drzewie\n");

printf("Przodkowie wezla: ");

printAncestors(found->parent);

} else {

printf("Wezel nie istnieje\n");

}

break;

case 3:

printf("Wezly w kolejnosci przechodzenia: ");

printInorder(root);

printf("\n");

break;

case 4:

printf("Wyjscie z programu\n");

break;

default:

printf("Zly wybor, sprobuj ponownie\n");

}

} while (choice != 4);

return 0;

}

3. Dane wejściowe /wyjściowe

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

1. Złożoność obliczeniowa algorytmu

O(n)