**SPRAWOZDANIE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Przedmiot** | Wprowadzenie do Informatyki | **Zadanie** | 1 |
| **Autor** | Mateusz Jasiński | **Grupa** | WCY20IY2S1 |
| **Temat** | Struktury dynamiczne – drzewa BST | | |

1. Treść

Napisz program, który umożliwi budowanie drzewa BST. Program powinien:  
-Umożliwić wprowadzanie danych z klawiatury, dodawanie węzła,  
-Wypisanie węzłów drzewa w kolejności przechodzenia LVR – in-order, przejście poprzeczne

* 1. Metoda realizacji

Wczytać wartość polecenie, w zależności od wartości:  
-Dodać węzeł,  
-Wypisać węzły w kolejności przechodzenia LVR.

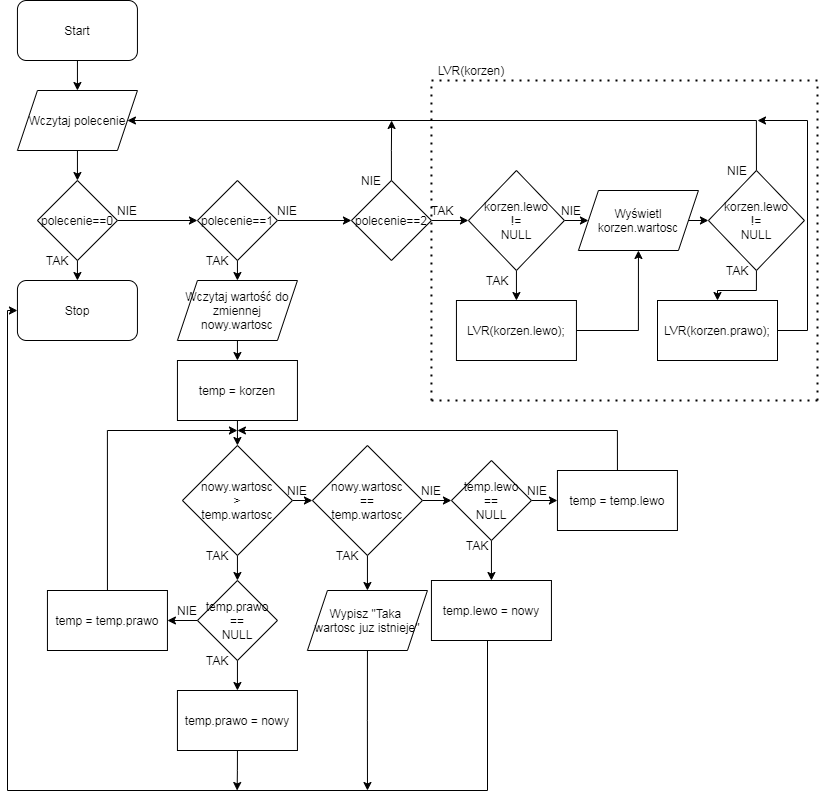
* 1. Założenia / ograniczenia dotyczące danych:
     1. Dane wejściowe

Wartości węzłów – wprowadzone z klawiatury

* + 1. Dane wyjściowe

Węzły w kolejności przechodzenia LVR – wyprowadzone na ekran

1. Realizacja
   1. Algorytm



* 1. Kod źródłowy

#include<stdio.h>

#include<stdlib.h>

typedef struct drzewo {

int wartosc;

struct drzewo \*lewo;

struct drzewo \*prawo;

} drzewo;

int gui() {

system("cls");

printf("Drzewo binarne\n"

"----------------------------------------------\n"

"[0] Wyjscie z programu\n"

"[1] Dodaj wezel\n"

"[2] Wypisz LVR\n"

"----------------------------------------------\n\n"

"Wybrana komenda: ");

}

Dodaj(drzewo \*\*korzen) {

drzewo \*nowy;

nowy=(drzewo\*)malloc(sizeof(drzewo));

nowy->lewo=NULL;

nowy->prawo=NULL;

printf("Podaj liczbe do wstawienia: ");

scanf("%d", &nowy->wartosc);

if(\*korzen==NULL)

\*korzen=nowy;

else {

drzewo \*temp=\*korzen;

while(temp) {

if(nowy->wartosc>temp->wartosc)

if(temp->prawo!=NULL)

temp=temp->prawo;

else {

temp->prawo=nowy;

break;

}

else if(nowy->wartosc<temp->wartosc)

if(temp->lewo!=NULL)

temp=temp->lewo;

else {

temp->lewo=nowy;

break;

}

else if(nowy->wartosc==temp->wartosc) {

printf("\nTaka wartosc jest juz w drzewie");

break;

}

}

}

printf("\nKliknij aby kontynuowac.");

getch();

}

LVR(drzewo \*korzen) {

if(korzen==NULL)

printf("Drzewo jest puste!");

else {

if(korzen->lewo!=NULL)

LVR(korzen->lewo);

printf("%d ",korzen->wartosc);

if(korzen->prawo!=NULL)

LVR(korzen->prawo);

}

}

main() {

drzewo \*korzen=NULL;

int polecenie;

do {

gui();

scanf("%d", &polecenie);

system("cls");

switch(polecenie) {

case 0:

printf("Zakonczono dzialanie programu.");

break;

case 1:

Dodaj(&korzen);

break;

case 2:

printf("Kolejnosc przechodzenia: ");

LVR(korzen);

printf("\nKliknij aby kontynuowac.");

getch();

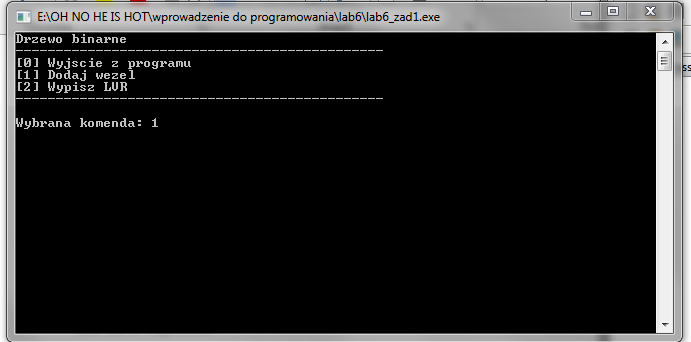
break;

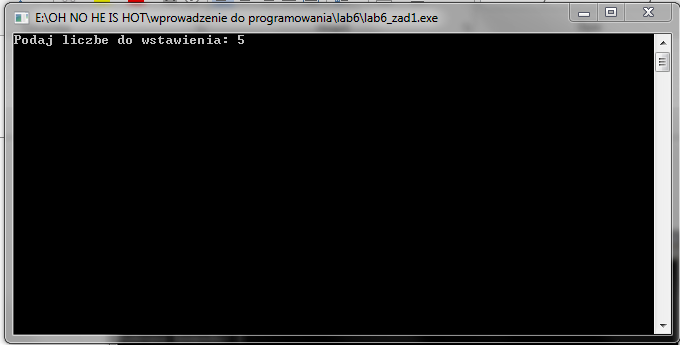
}

} while(polecenie);

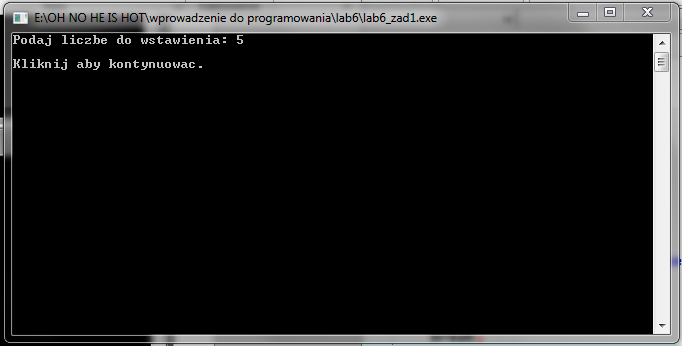
}

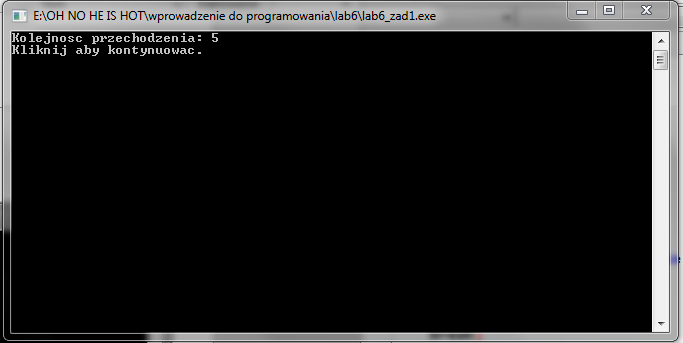
* 1. Dane wejściowe





* 1. Dane wyjściowe





1. Wnioski

O(n) – złożoność obliczeniowa jest liniowa i zależy od liczby wprowadzonych węzłów.