

SPRAWOZDANIE

Paweł Dzedzic

INŻYNIERIA I ANALIZA DANYCH

Nr albumu: 169774

Contents

1 Wstęp	1
2 Treść zadania	2
3 Opis programu.....	2
3.1 Struktura Node	2
3.2 Funkcja Main	2
3.3 Funkcja StworzWezel.....	3
3.4 Funkcja Dodaj	3
3.4.1 Pseudokod Dodaj.....	3
3.5 Funkcja Szukaj	4
3.5.1 Pseudokod Szukaj	4
3.6 Funkcja Min oraz Maks.....	5
3.6.1 Pseudokod Min oraz Maks	5
3.7 Funkcja Zlicz.....	6
3.7.1 Pseudokod Zlicz	6
3.8 Funkcja UsunWezel	6
3.8.1 Funkcja MinUsun	6
3.8.2 Pseudokod UsunWezel	8
3.9 Funkcja UsunDrzewo	9
3.9.1 Pseudokod UsunDrzewo.....	9
3.10 Funkcja WypiszDrzewo.....	10
3.10.1 Psuedokod WypiszDrzewo	10
4. Działanie programu:	11
5. Testy funkcji programu.....	13

1 Wstęp

Dokument jest sprawozdaniem z drugiego projektu z przedmiotu Algorytmy i struktury danych. Program został napisany w języku C++ w kompilatorze GNU GCC Compiler w programie Code::Blocks 20.03.

2 Treść zadania

Dokonaj implementacji struktury danych typu drzewo binarne wraz z wszelkimi potrzebnymi operacjami charakterystycznymi dla tej struktury (inicjowanie struktury, dodawanie/usuwanie elementów, wyświetlanie elementów, zliczanie/wyszukiwanie zadanego elementu itp.)

- przyjąć że podstawowym typem danych przechowywanym w elemencie struktury będzie struktura z jednym polem typu int
- w funkcji main() przedstawić możliwości napisanej przez siebie biblioteki
- kod powinien być opatrzony stosownymi komentarzami

3 Opis programu

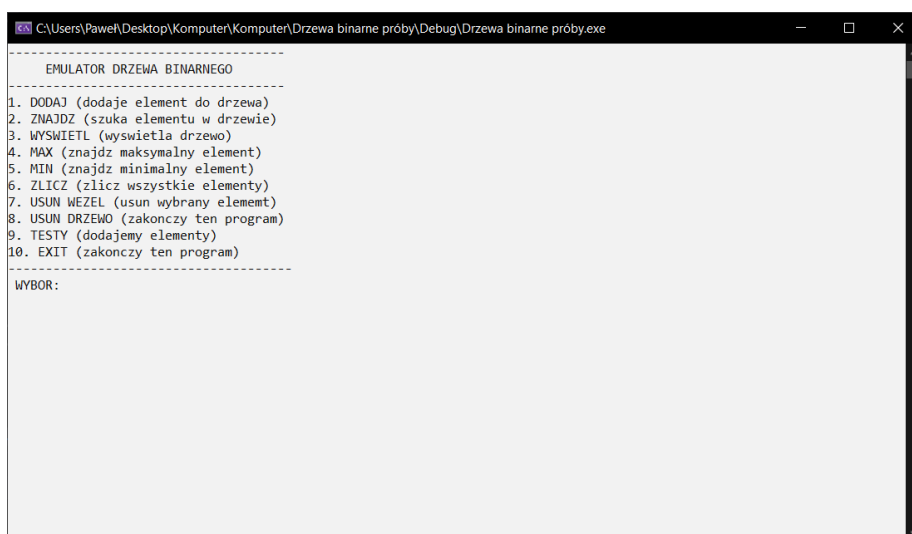
Program składa się z stworzonej struktury oraz kilku funkcji wywoływanych w głównej funkcji main()

3.1 Struktura Node

W celu inicjalizacji struktury drzewa binarnego, została utworzona struktura Node w której zawarte są trzy zmienne. Zmienna typu int przechowująca wartość danego węzła w drzewie oraz dwie zmienne typu struktury Node, które pozwalają na przypisanie wartości do danego węzła oraz poruszanie się po nich.

3.2 Funkcja Main

Funkcja main() składa się z kilku zmiennych pomocniczych, które są przydatne przy wykonywaniu operacji na drzewie oraz z zmiennej „root”, której został przypisany typ struktury Node. Wykonywana jest w środku funkcji, pętla while w której znajduje się instrukcja switch tworząca pewnego rodzaju menu, które inicjalizuje operacje na drzewie binarnym.



```

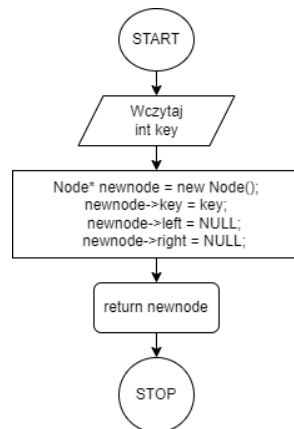
C:\Users\Paweł\Desktop\Komputer\Komputer\Drzewa binarne próby\Debug\Drzewa binarne próby.exe
-----
EMULATOR DRZEWY BINARNEGO
-----
1. DODAJ (dodaje element do drzewa)
2. ZNAJDZ (szuka elementu w drzewie)
3. WYŚWIETL (wyświetla drzewo)
4. MAX (znajdź maksymalny element)
5. MIN (znajdź minimalny element)
6. ZLICZ (zlicz wszystkie elementy)
7. USUN WEZEL (usun wybrany element)
8. USUN DRZEWO (zakńczy ten program)
9. TESTY (dodajemy elementy)
10. EXIT (zakńczy ten program)
-----
WYBOR:

```

W zależności od wyboru liczby od 1 do 9, funkcja main() wywołuje poszczególne funkcje znajdujące się w kodzie programu.

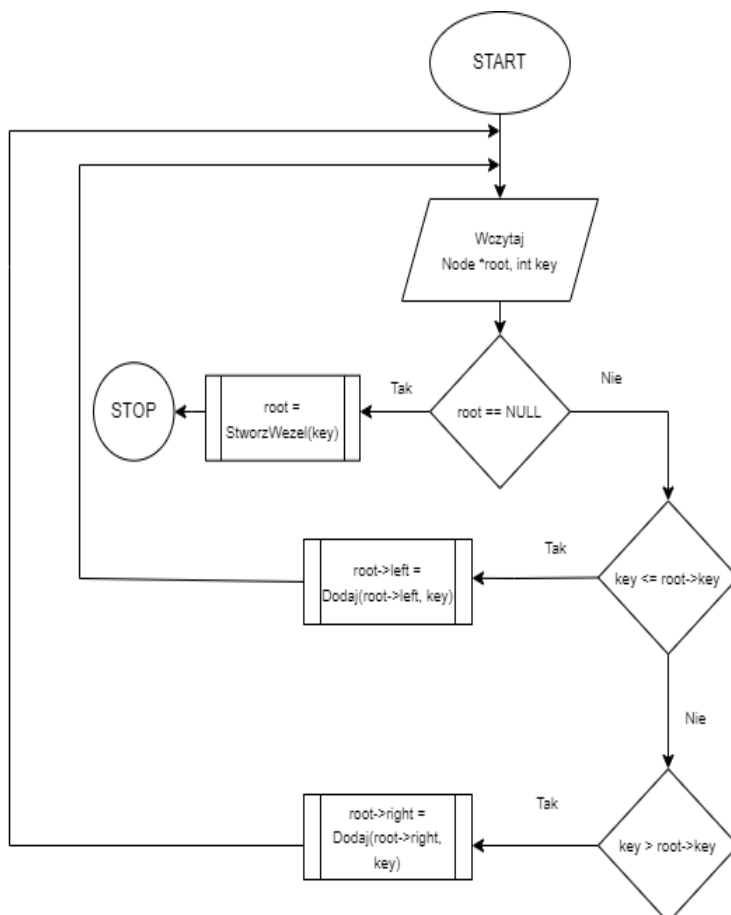
3.3 Funkcja StworzWezel

W funkcji StworzWezel() tworzony jest nowy węzeł, któremu nadawana jest wybrana wartość oraz tworzone są dodatkowo dwa węzły a prawej i lewej strony przyjmujące wartość NULL.



3.4 Funkcja Dodaj

W funkcji Dodaj() dodawany jest węzeł poprzez przeszukanie całego drzewa i znalezienie w nim miejsca dla nowego węzła.



3.4.1 Pseudokod Dodaj

Wczytaj Node* root

Wczytaj int key

Jeśli root == NULL to

Wywołaj i przypisz

root = StworzWezel(key)

Zakończ

Jeśli key <= root->key

Wywołaj

Dodaj(root->left, key)

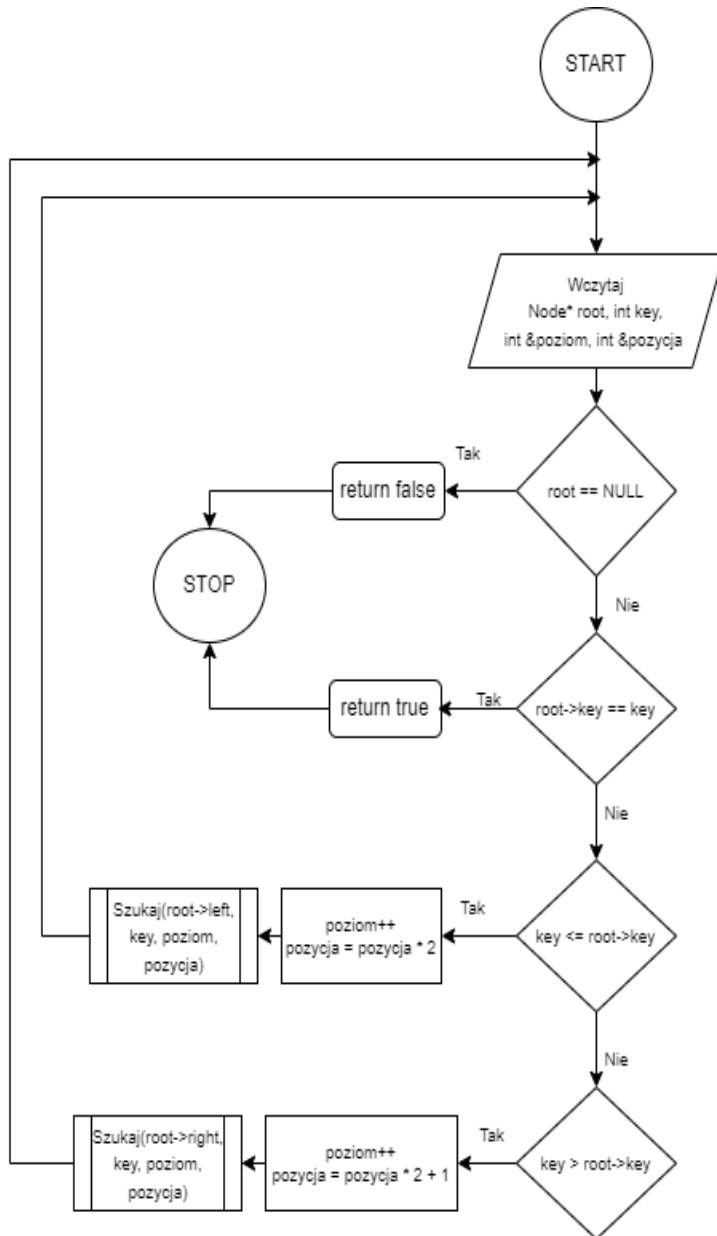
Jeśli key > root->key

Wywołaj

Dodaj(root->right, key)

3.5 Funkcja Szukaj

Funkcja Szukaj() przeszukuje całe drzewo i gdy znajdzie szukany element zwraca true, w przeciwnym razie zwraca false. Zwracane są również dwie zmienne pomocnicze które wskazują poziom i pozycję na której znajduje się szukany element.



3.5.1 Pseudokod Szukaj

Wczytaj Node* root

Wczytaj int key, &poziom, &pozycja

Jeśli root == NULL to

Zwróć fałsz

Zakończ

Jeśli root->key == key to

Zwróć prawdę

Zakończ

Jeśli key <= root->key

Inkrementuj poziom

pozycja <- pozycja * 2

Wywołaj

Szukaj(root->left, key, poziom, pozycja)

Jeśli key > root->key

Inkrementuj poziom

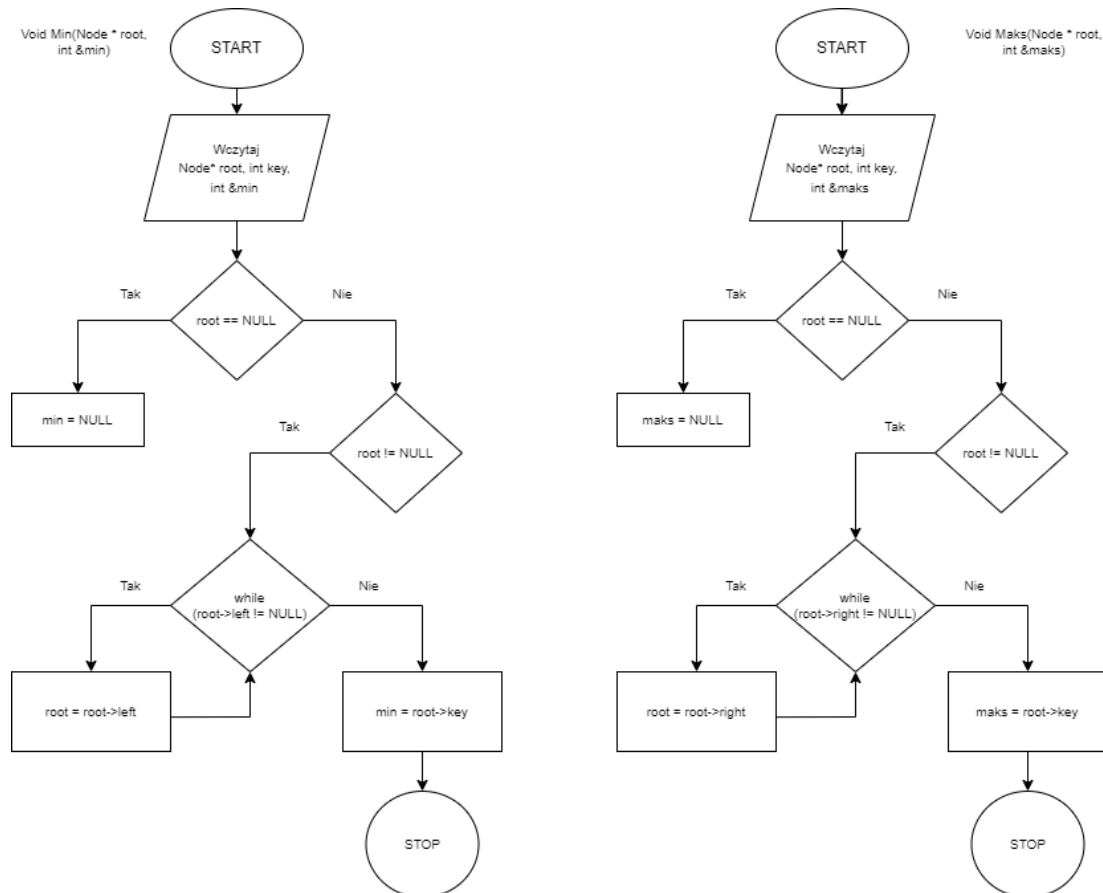
pozycja <- pozycja * 2

Wywołaj

Szukaj(root->right, key, poziom, pozycja)

3.6 Funkcja Min oraz Maks

Funkcje Min() oraz Maks() przesuwają się analogicznie po lewej gałęzi, aby znaleźć węzeł z minimalną wartością oraz po prawej, aby znaleźć z węzeł z maksymalną wartością. Za pomocą referencji wartości są zwracane do funkcji main()



3.6.1 Pseudokod Min oraz Maks

Wczytaj Node* root

Wczytaj int key, &maks

Jeśli root == NULL to

Przypisz maks <- NULL

Jeśli root != NULL to

Dopóki root->right != NULL wykonuj

root <- root->right

Jeśli root->right == NULL to

Przypisz maks <- root->key

Zakończ

Wczytaj Node* root

Wczytaj int key, &min

Jeśli root == NULL to

Przypisz min <- NULL

Jeśli root != NULL to

Dopóki root->left != NULL wykonuj

root <- root->left

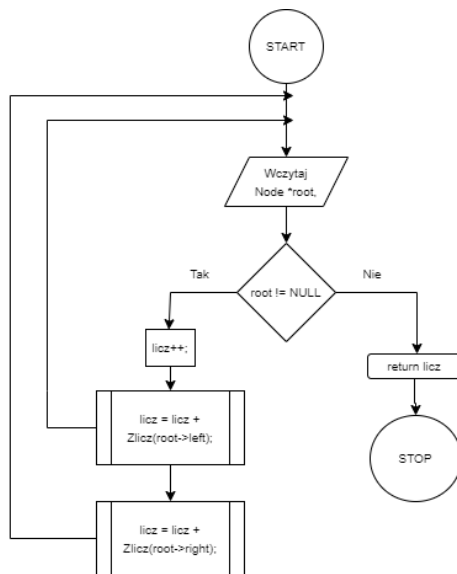
Jeśli root->left == NULL to

Przypisz min <- root->key

Zakończ

3.7 Funkcja Zlicz

Funkcja Zlicz() zlicza ilość wszystkich węzłów znajdujących się w drzewie.



3.7.1 Pseudokod Zlicz

Wczytaj *Node* root*

Wczytaj *int key*

Stwórz *int licz*

Jeśli *root != NULL* to

Inkrementuj *licz*

Wywołaj i przypisz *licz <- licz + Zlicz(root->left)*

Wywołaj i przypisz *licz <- licz + Zlicz(root->right)*

Zwróć *licz*

Zakończ

3.8 Funkcja UsunWezel

Funkcja UsunWezel() pozwala na usunięcie wybranego węzła po podaniu jego wartości. Odnajduje podaną wartość węzła (o ile znajduje się ona w drzewie) i następnie wybierając spośród trzech przypadków usuwa węzeł.

3.8.1 Funkcja MinUsun

Funkcja MinUsun() pozwala na znalezienie najmniejszego węzła od węzła usuwanego, aby nadpisać jego wartość, wartością szukanego minimalnego węzła (jest tylko wywoływana w funkcji UsunWezel() w przypadku gdy węzeł posiada dwójkę dzieci).

3.8.1.1 Pseudokod MinUsun

Wczytaj *Node* root*

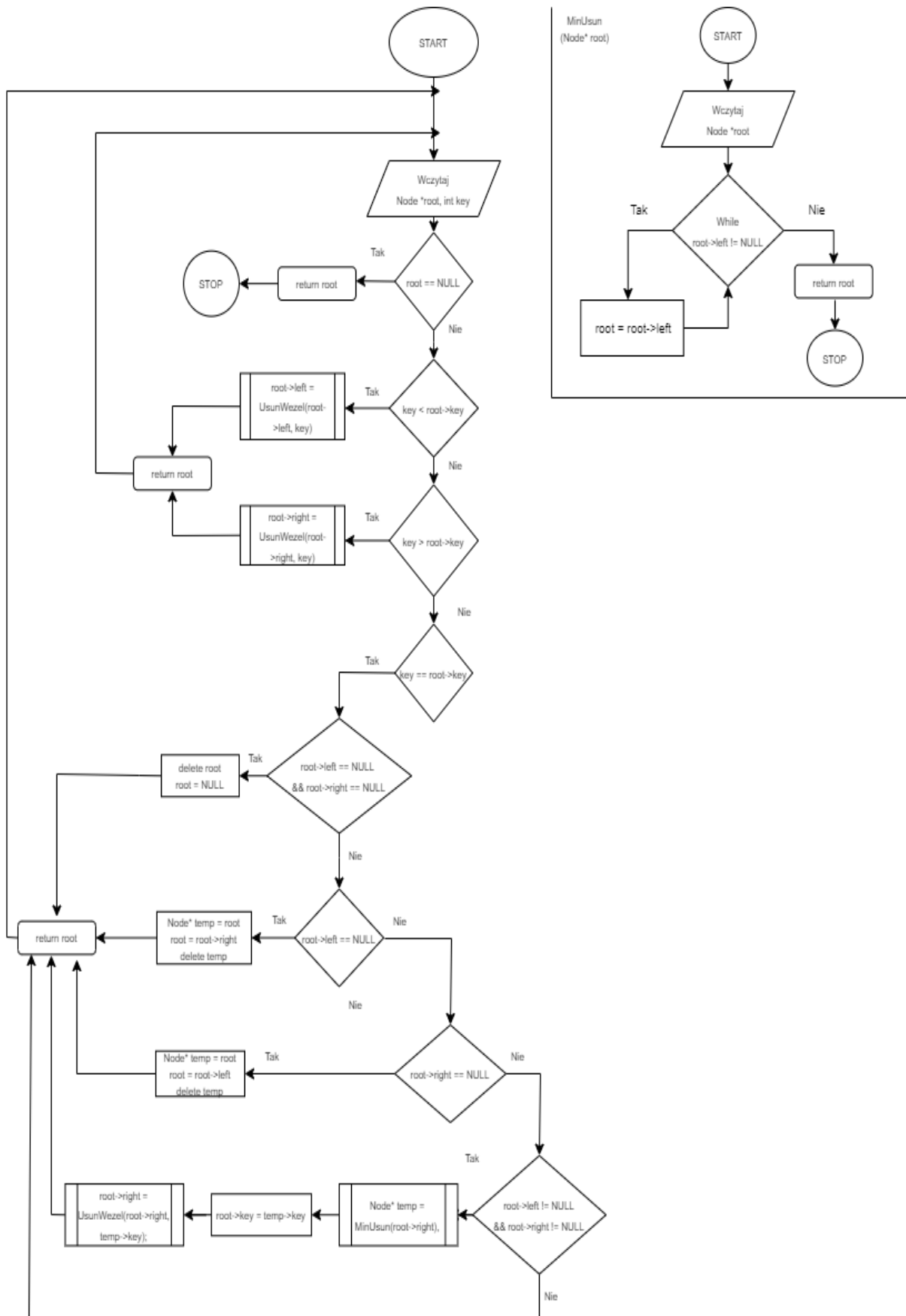
Dopóki *root->left != NULL*

root <- root->left

Jeśli *root->left != NULL*

Zwróć *root*

Zakończ



3.8.2 Pseudokod UsunWezel

Wczytaj *Node** root

Wczytaj *int* key

Jeśli *root == NULL* to

 Zwróć *root*

 Zakończ

Jeśli *key < root->key* to

 Wywołaj i przypisz *root->left <- UsunWezel(root->left, key)*

 Zwróć *root*

Jeśli *key > root->key* to

 Wywołaj i przypisz *root->right <- UsunWezel(root->right, key)*

 Zwróć *root*

Jeśli *key == root->key* to

 Sprawdź

 Jeśli *root->left == NULL* oraz *root->right == NULL* to

 Usuń *root*

root <- NULL

 Zwróć *root*

 Jeśli *root->left* to

 Stwórz *Node** temp

temp <- root

root <- root->right

 Usuń *temp*

 Zwróć *root*

 Jeśli *root->right* to

 Stwórz *Node** temp

temp <- root

root <- root->left

 Usuń *temp*

 Zwróć *root*

 Jeśli *root->left != NULL* oraz *root->right != NULL* to

 Stwórz *Node** temp

 Wywołaj i przypisz *temp <- MinUsun(root->right)*

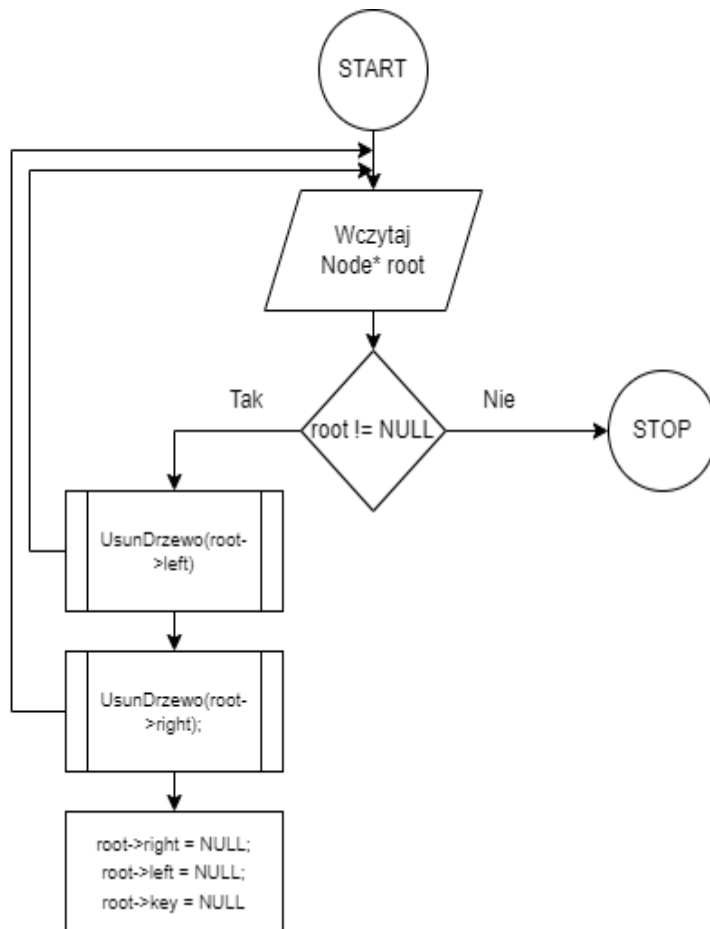
root->key <- temp->key

 Wywołaj *UsunWezel(root->right, temp->key)*

 Zwróć *root*

3.9 Funkcja UsunDrzewo

Funkcja UsunDrzewo() pozwala na usunięcie wszystkich węzłów w drzewie wraz z ich wartościami



3.9.1 Pseudokod UsunDrzewo

Wczytaj Node* root

Jeśli root!=NULL

UsunDrzewo(root->left)

UsunDrzewo(root->right)

root->right <- NULL

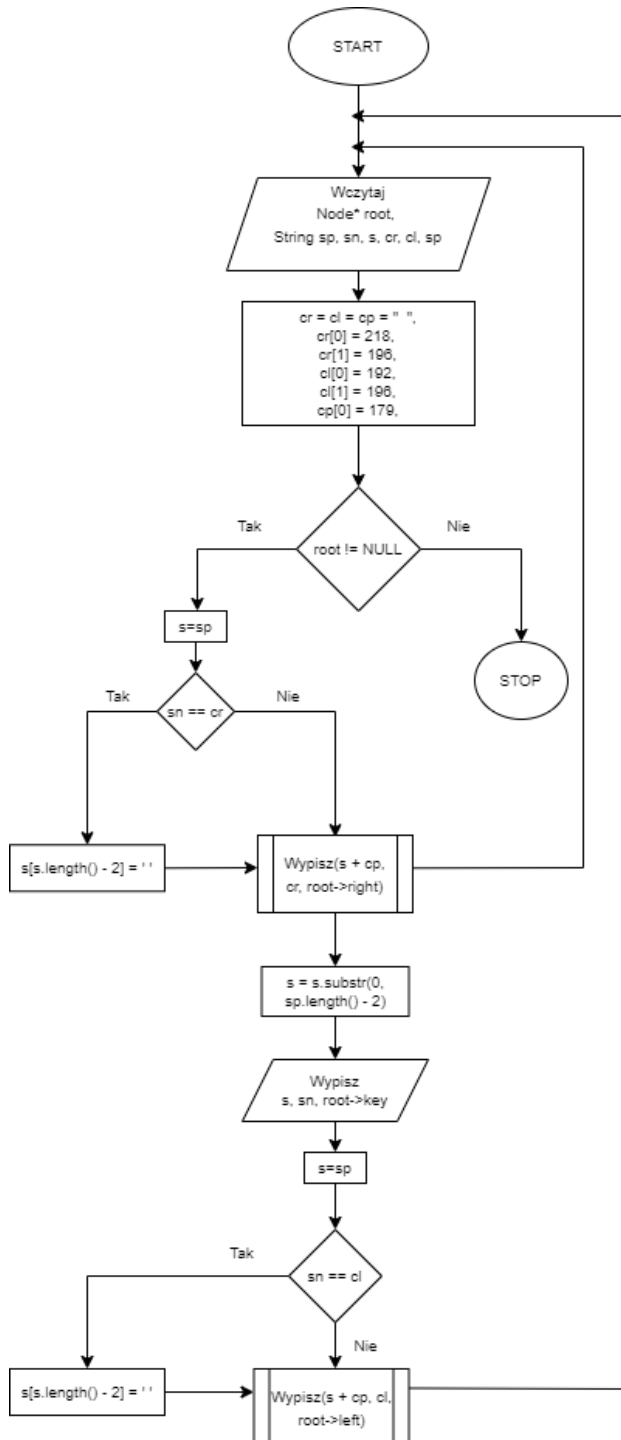
root->left <- NULL

root->key <- NULL

Zakończ

3.10 Funkcja WypiszDrzewo

Funkcja WypiszDrzewo() wypisuje całą strukturę drzewa wraz z jej elementami.



3.10.1 Psuedokod WypiszDrzewo

Wczytaj string *sp*, *sn*

Wczytaj *Node* root*

Stwórz string *s*, *cr*, *cl*, *sp*

cr <- *cl* <- *cp* <- " "

cr[0] <- "r"

cl[0] <- "L"

cp[0] <- "|"

cr[1] <- "_"

cl[1] <- "_"

Jeśli *root != NULL* wykonuj

s <- *sp*

Jeśli *sn* = "r" to

s [*| s |* - 2] <- " "

Wypisz(*s* + "|" , "r" , *root->right*)

s <- *sp.substr*[0, *|s|* - 2]

Pisz *s*, *sn*, *root->key*

s <- *sp*

Jeśli *sn* = "L" to

s [*| s |* - 2] <- " "

Wypisz(*s* + "|" , "L" , *root->left*)

Zakończ

4. Działanie programu:



```
C:\Users\Paweł\Desktop\Komputer\Komputer\Drzewa binarne próby\Debug\Drzewa binarne próby.exe

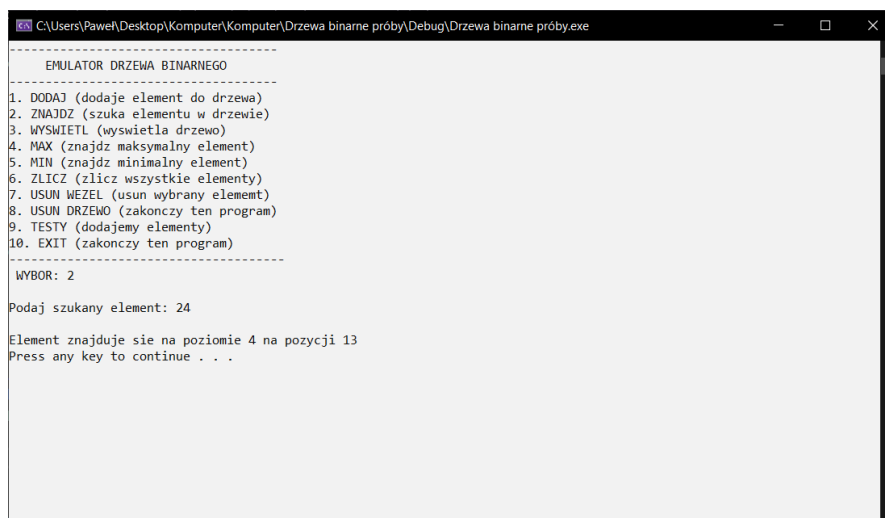
-----
EMULATOR DRZEWIA BINARNEGO
-----
1. DODAJ (dodaje element do drzewa)
2. ZNAJDZ (szuka elementu w drzewie)
3. WYŚWIETL (wyswietla drzewo)
4. MAX (znajdz maksymalny element)
5. MIN (znajdz minimalny element)
6. ZLICZ (zlicz wszystkie elementy)
7. USUN WĘZEL (usun wybrany element)
8. USUN DRZEWO (zakonczy ten program)
9. TESTY (dodajemy elementy)
10. EXIT (zakonczy ten program)
-----

WYBOR: 3

      38
     / \
    33  33
   / \
  32  24
 / \
9  21
/ \
8  4
/ \
2  2

Press any key to continue . . .
```

Rysunek 1: Wyswietl drzewo



```
C:\Users\Paweł\Desktop\Komputer\Komputer\Drzewa binarne próby\Debug\Drzewa binarne próby.exe

-----
EMULATOR DRZEWIA BINARNEGO
-----
1. DODAJ (dodaje element do drzewa)
2. ZNAJDZ (szuka elementu w drzewie)
3. WYŚWIETL (wyswietla drzewo)
4. MAX (znajdz maksymalny element)
5. MIN (znajdz minimalny element)
6. ZLICZ (zlicz wszystkie elementy)
7. USUN WĘZEL (usun wybrany element)
8. USUN DRZEWO (zakonczy ten program)
9. TESTY (dodajemy elementy)
10. EXIT (zakonczy ten program)
-----


WYBOR: 2

Podaj szukany element: 24

Element znajduje sie na poziomie 4 na pozycji 13

Press any key to continue . . .
```

Rysunek 2: Znajdz element



```
C:\Users\Paweł\Desktop\Komputer\Komputer\Drzewa binarne próby\Debug\Drzewa binarne próby.exe

-----
EMULATOR DRZEWIA BINARNEGO
-----
1. DODAJ (dodaje element do drzewa)
2. ZNAJDZ (szuka elementu w drzewie)
3. WYŚWIETL (wyswietla drzewo)
4. MAX (znajdz maksymalny element)
5. MIN (znajdz minimalny element)
6. ZLICZ (zlicz wszystkie elementy)
7. USUN WĘZEL (usun wybrany element)
8. USUN DRZEWO (zakonczy ten program)
9. TESTY (dodajemy elementy)
10. EXIT (zakonczy ten program)
-----

WYBOR: 6

Drzewo posiada 10 węzłow

Press any key to continue . . .
```

Rysunek 3: Zlicz węzły

```
C:\Users\Paweł\Desktop\Komputer\Komputer\Drzewa binarne próby\Debug\Drzewa binarne próby.exe

-----
EMULATOR DRZEWA BINARNEGO
-----
1. DODAJ (dodaje element do drzewa)
2. ZNAJDZ (szuka elementu w drzewie)
3. WYSWIETL (wyswietla drzewo)
4. MAX (znajdz maksymalny element)
5. MIN (znajdz minimalny element)
6. ZLICZ (zlicz wszystkie elementy)
7. USUN WEZEL (usun wybrany element)
8. USUN DRZEWO (zakonczy ten program)
9. TESTY (dodajemy elementy)
10. EXIT (zakonczy ten program)
-----

WYBOR: 4

Maksymalna wartosc elementu w drzewie wynosi: 38
Press any key to continue . . .
```

Rysunek 4: Wyświetl maks

```
C:\Users\Paweł\Desktop\Komputer\Komputer\Drzewa binarne próby\Debug\Drzewa binarne próby.exe

-----
EMULATOR DRZEWA BINARNEGO
-----
1. DODAJ (dodaje element do drzewa)
2. ZNAJDZ (szuka elementu w drzewie)
3. WYSWIETL (wyswietla drzewo)
4. MAX (znajdz maksymalny element)
5. MIN (znajdz minimalny element)
6. ZLICZ (zlicz wszystkie elementy)
7. USUN WEZEL (usun wybrany element)
8. USUN DRZEWO (zakonczy ten program)
9. TESTY (dodajemy elementy)
10. EXIT (zakonczy ten program)
-----

WYBOR: 3

      38
     /  \
    33   24
   /  \  /  \
  33  21 8   4
 /  \ /  \
9   2 4   2
/  \
2   2

Press any key to continue . . .
```

Rysunek 5: Drzewo po usunięciu elementu 170

5. Testy funkcji programu

Testy były wykonywane na funkcji Dodaj() oraz WypiszDrzewo(). Elementy były losowane a następnie dodawane do drzewa. Przedział losowanych liczb był wprost proporcjonalny do rozmiaru tablicy. Czas jest wyrażony w milisekundach (10^{-3} s).

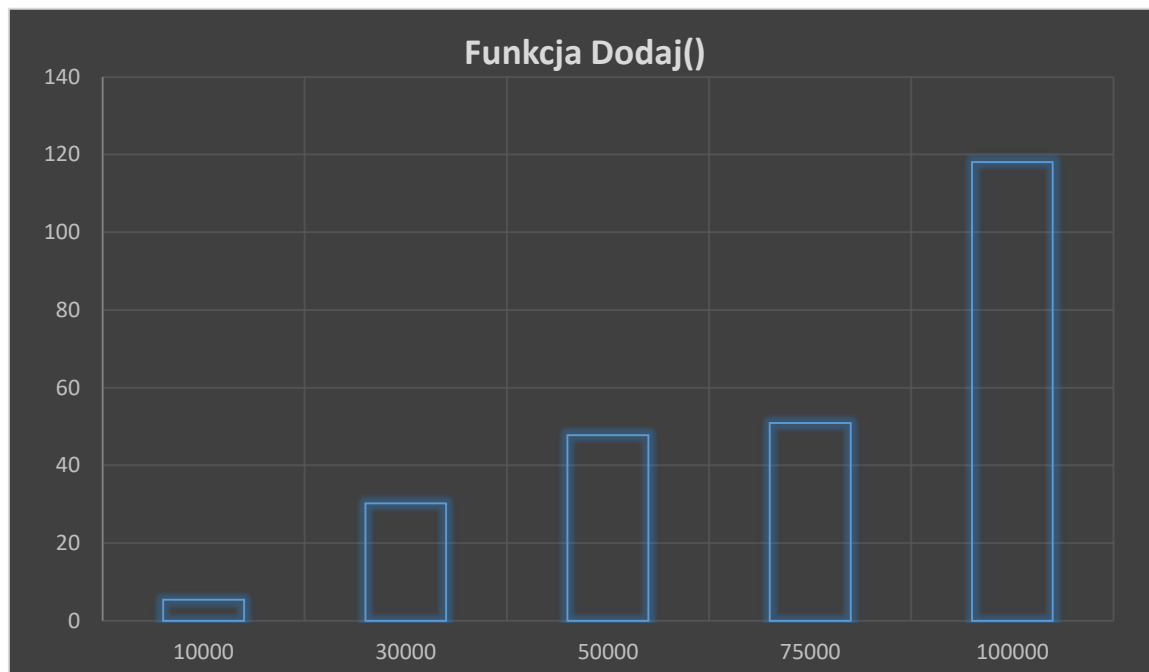


Tabela 1: Wykres czasu dla funkcji dodaj

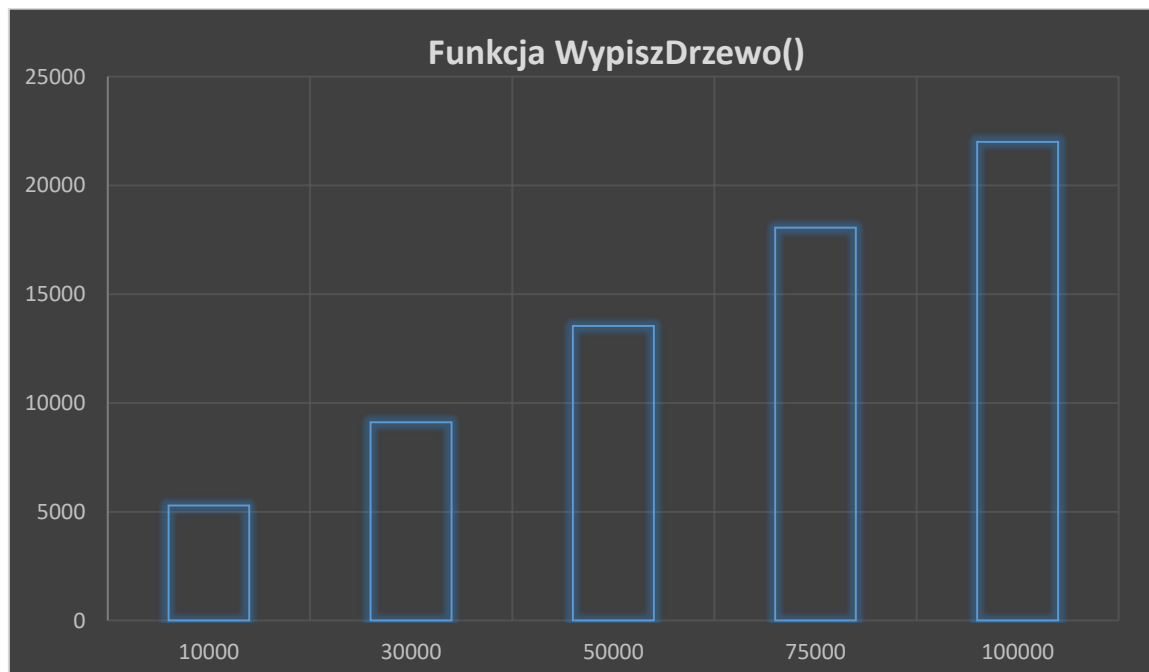


Tabela 2: Wykres czasu dla funkcji WypiszDrzewo()

