МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Алгоритмы и структуры данных»

Тема: Вычисление высоты дерева

Студент гр. 1303 _____ Чубан Д.В.

Преподаватель _____ Иванов Д. В.

Санкт-Петербург

2022

Цель работы.

Научиться итеративно обходить дерево, задаваемое списком родительских вершин. Основываясь на данном алгоритме, написать программу, вычисляющую высоту дерево, а так же написать тесты для данной программы.

Задание.

Вычисление высоты дерева.

На вход программе подается корневое дерево с вершинами $\{0, \ldots, n-1\}$, заданное как последовательность $parent_0, \ldots, parent_{n-1}$, где $parent_i$ – родитель i-й вершины. Требуется вычислить и вывести высоту этого дерева.

Формат входа.

Первая строка содержит натуральное число n. Вторая строка содержит n целых чисел $parent_0, \ldots, parent_{n-1}$. Для каждого $0 \le i \le n - 1$, $parent_i$ — родитель вершины i; если parent $i = -1$, то i является корнем. Гарантируется, что корень ровно один и что данная последовательность задаёт дерево.

Формат выхода.

Высота дерева.

Выполнение работы.

Для вычисления длины дерева, заданного списком его родительских вершин была реализована функции *main()* и *findLen(int n, int* maxLen, int* tree).*

В функции *main()* производится считывание входных данных. Далее для каждого элемента введенного списка применяется функция *findLen(),* которая идет от переданного в нее элемента дерева к его корню и считает количество пройденных элементов. После того, как функция дошла до корня дерева, текущее количество сравнивается с максимально ранее найденным, и если оно превосходит его, то переменная с максимумом обновляется.

Функция выводит найденный максимум.

Разработанный программный код см. в Приложении А.

Тестирование.

Код файла с тестами содержится в приложении Б.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	-2 []	Invalid value	Значения n <= 0 недопустимы
2.	0 []	Invalid value	Значения n <= 0 недопустимы
3.	5 4 -1 4 1 1	3	
4.	5 -1 0 4 0 3	4	

Выводы.

В ходе лабораторной работы была написана программа вычисляющая высоту дерева, заданного списком родительских вершин. Так же все написанные функции были покрыты тестами, проверяющими корректность работы алгоритмов при стандартных, пограничных и нестандартных случаях.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include "modules/findLen.hpp"

using namespace std;

int main(){
int n;
cin >> n;
if (n <= 0){
cout << "Invalid value" << endl;
};
int maxLen = 0;
int* tree = new int[n];
for (int i = 0; i < n; i++){
	int temp;
	cin >> temp;
	tree[i] = temp;
}
for (int i = 0; i < n; i++){
	if(tree[i+1] != i){
		findLen(i, &maxLen, tree);
	}
}
cout << maxLen << endl;
return 0;
}
```

Название файла: findLen.cpp

```
#include "findLen.hpp"
#include <iostream>

using namespace std;

int findLen(int n, int* maxLen, int* tree){
	if (n < -1){
		return FAIL_STATUS;
	}
	if(sizeof(tree)/sizeof(int) == 0){
		return FAIL_STATUS;
	};
	int tempLen = 0;
	while (n != -1){
		n = tree[n];
		tempLen++;
		if (tempLen > *maxLen){
```

```
                        *maxLen = tempLen;
                }
        }
        if(*maxLen == 0){
                *maxLen = 1;
        }
        return SUCCES_STATUS;
}
```

ПРИЛОЖЕНИЕ Б

ФАЙЛ ТЕСТИРОВАНИЯ ПРОГРАММЫ

Название файла: test.cpp

```
#include <cassert>

#include <iostream>
#include "modules/findLen.hpp"

using namespace std;

void testSizeNeg(){
	int n = -2;
	int* tree = new int[1];
	int maxLen = 0;
	assert(findLen(n, &maxLen, tree) == FAIL_STATUS);
}

void testSizeOne(){
	int n = 0;
	int* tree = new int[1];
	tree[0] = -1;
	int maxLen = 0;
	assert(findLen(n, &maxLen, tree) == SUCCES_STATUS);
}

void testMoodle1(){
	int n = 5;
	int* tree = new int[5];
	tree[0] = 4;
	tree[1] = -1;
	tree[2] = 4;
	tree[3] = 1;
	tree[4] = 1;
	int maxLen = 0;
	for (int i = 0; i < n; i++){
		if(tree[i+1] != i){
			findLen(i, &maxLen, tree);
		}
	}
	assert(maxLen == 3);
}

void testMoodle2(){
	int n = 5;
	int* tree = new int[5];
	tree[0] = -1;
	tree[1] = 0;
	tree[2] = 4;
	tree[3] = 0;
	tree[4] = 3;
```

```
	int maxLen = 0;
	for (int i = 0; i < n; i++){
		if(tree[i+1] != i){
			findLen(i, &maxLen, tree);
		}
	}
	assert(maxLen == 4);
}

int main(){
	testSizeNeg();
	testSizeOne();
	testMoodle1();
	testMoodle2();
	return 0;
}
```