

Санкт-Петербургский политехнический университет Петра Великого  
Институт компьютерных наук и технологий  
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**Отчёт по лабораторной работе № 5**

Дисциплина: Низкоуровневое программирование

Тема: Программирование на языке С

Вариант: 1

Выполнил студент гр. 3530901/90002 \_\_\_\_\_ Д. Е. Бакин  
(подпись)

Принял старший преподаватель \_\_\_\_\_ Д. С. Степанов  
(подпись)

“ \_\_\_\_ ” \_\_\_\_\_ 2021 г.

Санкт-Петербург  
2021

## Постановка задачи:

1. Разработать статическую библиотеку, реализующую определенный вариант задания абстрактный тип данных.
2. Разработать демонстрационную программу – консольное приложение, обеспечивающее ввод данных из файла (файлов), их обработку и вывод в файл (файлы); имена файлов передаются в качестве параметров командной строки.

## Требования к ПО:

1. Язык разработки – С.
2. Реализация абстрактного типа данных должна использовать динамическое выделение памяти, при этом должна быть предусмотрена функция деинициализации, обеспечивающая освобождение всей выделенной памяти.
3. Библиотека и демонстрационная программа должны быть снабжены модульными тестами.
4. Разработанный исходный код должен компилироваться gcc без ошибок и предупреждений со следующими параметрами: `-std=c11 -pedantic -Wall -Wextra`.
5. Сборка библиотеки, демонстрационной программы и модульных тестов должна осуществляться утилитой `make`.

## Вариант:

Дерево (с узлами произвольной степени). Каждый узел может иметь разное количество потомков. Содержимое узла - целое число.

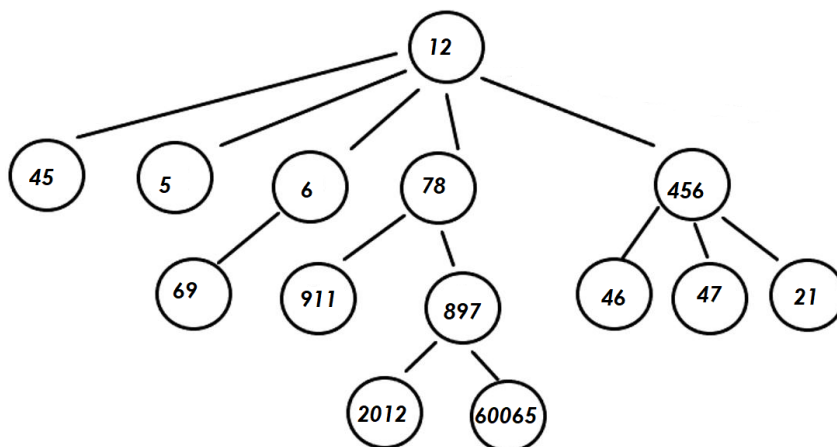


Рис. 1 Пример дерева с узлами произвольной степени.

## Описание API библиотеки:

Дерево было реализованно по динамической структуре узла:

*int data* – значение узла

*Node \*down* – указатель на самого левого ребенка

*Node \*right* – указатель на следующий узел с таким же родителем

Структура дерева содержит только ссылку на корень дерева (узел).

Библиотека имеет следующие функции:

*struct tree create(int x);* - создание корневого узла с заданным значением.

*Node \*addNode(int x, Node \*tree);* - добавление ребенка к узлу.

*void removeNode(Node \*node, Tree \*tree);* - удаление узла по ссылке в дереве

вместе со всеми его потомками.

*Node \*findNode(int x, Tree \*tree);* - нахождение узла в дереве с определенным параметром.

*Node \*findParent(Node \*node, Tree \*tree);* - нахождение родителя узла в дереве.

*void tprint(Tree \*tree);* - вывод дерева в консоль.

*int calcChild(Node \*node);* - подсчет количества детей узла.

*Node \*findMax(Tree \*tree);* - поиск узла в дереве с максимальным значением.

## Консольное приложение:

Представляет собой демонстрацию функционала библиотеки в виде двух функций. Для использования программы, нужно собрать программу, а после прописать `make app`.

Сначала идет сбор библиотеки, потом сбор приложения, после чего появляется открываемое приложение.

В приложении можно вручную задать дерево, и посмотреть, как работают различные функции.

```

Where connect next node?7358
-----Tree-----
      15
       7358
         1618
         27812
         22829
-----
Where connect next node?7358
-----Tree-----
      15
       7358
         1618
         20566
         27812
         22829
-----
Where connect next node? 22829
-----Tree-----
      15
       7358
         1618
         20566
         27812
         22829
         16607
-----

```

Рис. 2 Пример создания дерева вручную.

```

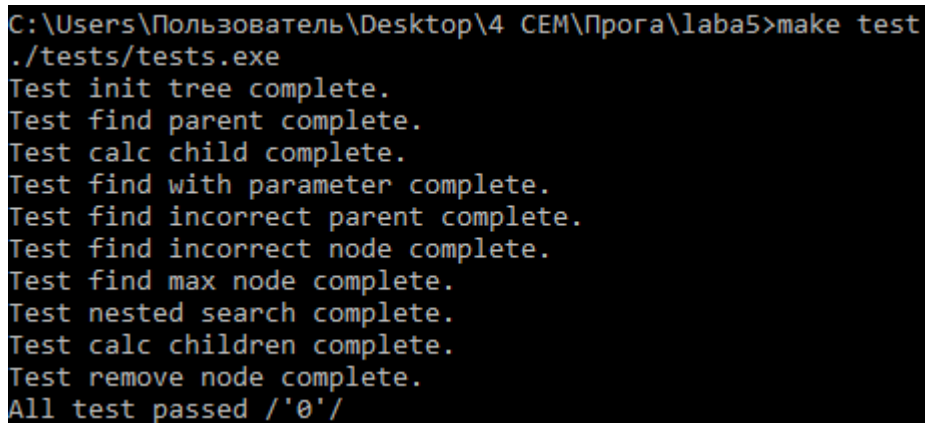
How many nodes want to delete: 5
Removing element: 55
-----Tree-----
      10
       228
        69
         98
         12
        47
       137
-----
Max element = 228
Removing element: 98
-----Tree-----
      10
       228
        69
         12
        47
       137
-----
Max element = 228
Removing element: 228
-----Tree-----
      10
       137
-----
Max element = 137
Removing element: 

```

Рис. 3 Пример удаления элементов из дерева.

## Модульные тесты:

Была написана библиотека тестов, и приложение для нее. Библиотека тестирует функции библиотеки на корректность. Запуск тестов можно осуществить командой `make test`, после сборки всей программы.



```
C:\Users\Пользователь\Desktop\4 СЕМ\Прора\laba5>make test
./tests/tests.exe
Test init tree complete.
Test find parent complete.
Test calc child complete.
Test find with parameter complete.
Test find incorrect parent complete.
Test find incorrect node complete.
Test find max node complete.
Test nested search complete.
Test calc children complete.
Test remove node complete.
All test passed /'0'/'
```

Рис. 4 Тестировка библиотеки.

## Руководство программиста:

Копируем репозиторий <https://github.com/donebd/lowproglabs/tree/main/laba5>

Далее в консоли работа с `make`-файлом в корневой папке проекта:

`make` – сборка библиотеки, приложения и тестов.

`make buildlib` – отдельная сборка библиотеки.

`make buildtest` – сборка библиотеки и приложения тестов.

`make buildapp` – сборка демонстрационного приложения.

`make test` – запуск тестов.

`make app` – запуск демонстрационного приложения.

`make clean` – очистка всех созданных файлов.

**Вывод:**

Была написана статически линкуемая библиотека дерева с произвольным количеством узлов. Также была написана демонстрационная программа, показывающая функционал библиотеки и библиотека модульных тестов с приложением тестирования. Написаны make-файлы для упрощения работы с библиотекой.