**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по курсовой работе**

**по дисциплине «Программирование»**

Тема: “Бинарное дерево поиска, АВЛ-Дерево”

|  |  |  |
| --- | --- | --- |
| Студент гр. 2372 |  | Васильев Ю.А. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2023

**Цель работы.**

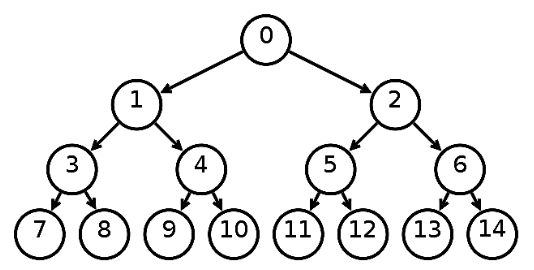
изучение свойств и организации АВЛ-Деревьев и деревьев бинарного поиска; получение практических навыков в работе с АВЛ и БП-деревьями; определение преимуществ и недостатков подобных структур данных; проведение сравнительной характеристики скорости вставки, удаления и поиска элемента в АВЛ и БП деревьях

**Основные теоретические положения.**

*Дерево бинарного поиска*

Бинарное дерево – это динамическая структура данных, состоящая из узлов, каждый из которых содержит кроме данных не более двух ссылок на различные бинарные деревья. На каждый узел имеется ровно одна ссылка. Начальный узел называется корнем дерева.

Узел, не имеющий поддеревьев, называется листом. Исходящие узлы называются предками, входящие – потомками. Высота дерева определяется количеством уровней, на которых располагаются его узлы.



Дерево является рекурсивной структурой данных, поскольку каждое поддерево также является деревом. Действия с такими структурами лучше всего описывать с помощью рекурсивных алгоритмов.

Бинарные дерева нужны:

Если необходимо расположить информацию, связанную между собой некой иерархией. Примером является файловая система компьютера.

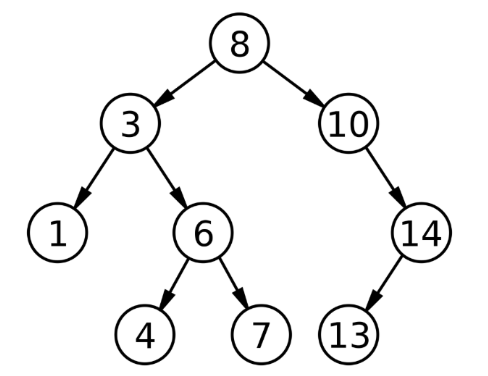
Если необходимо хранить данные, составленные в виде определённой структуры. Тогда хранение в виде бинарного дерева позволяет уменьшить скорость поиска данных и доступа к хранимой информации.

Если необходима высокая скорость добавления или удаления данных.

Если заранее неизвестен хранимый объем данных. Бинарные деревья, также, как и связанные списки, не имеют ограничения на количество узлов, поскольку узлы связаны указателями.

Бинарное дерево поиска – бинарное дерево, которое обладает дополнительными свойствами. Если дерево организовано таким образом, что для каждого узла все ключи левого поддерева меньше ключа этого узла, а все ключи его правого поддерева больше, то его можно называть бинарным деревом поиска.

Иными словами, данные в бинарном дереве поиска хранятся в отсортированном виде. Одинаковые ключи не допускаются.



При вставке элемента в бинарное дерево поиска необходимо учитывать два случая:

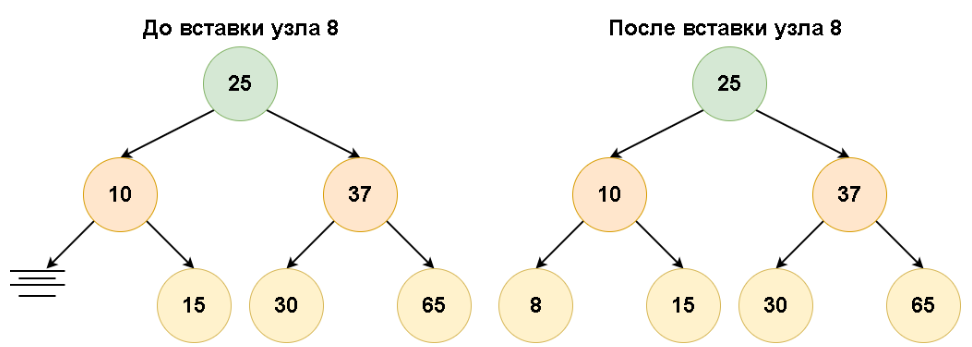
Дерево пустое;

Дерево непустое.

Если дерево пустое, то создается новый узел, который добавляется в дерево. Если дерево не пустое, то значение ключа добавляемого элемента сравнивается со значением ключа в узле, начиная от корня.

Так как одинаковые ключи не допускаются, перед вставкой нового элемент нужно проверить, нет ли уже в бинарном дереве такого элемента. Для этого необходимо начать обход дерева с корневого узла и проверить, не превосходит ли значение корневого узла добавляемого значения.

Если корневой узел больше добавляемого элемента, то необходимо переместиться в левое дочернее дерево. В противном случае – в правое. После выполнения данной проверки можно переходить к добавлению узла в бинарное дерево.



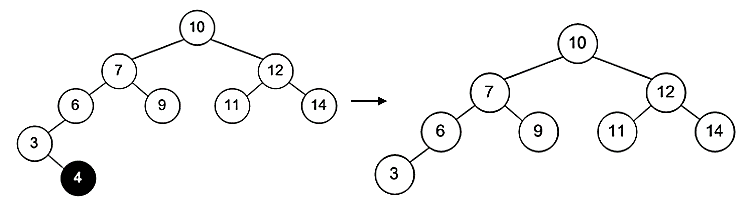
С удалением узла дело обстоит немного иначе. Существует несколько возможных ситуаций:

1) у узла нет наследников (узел является листом);

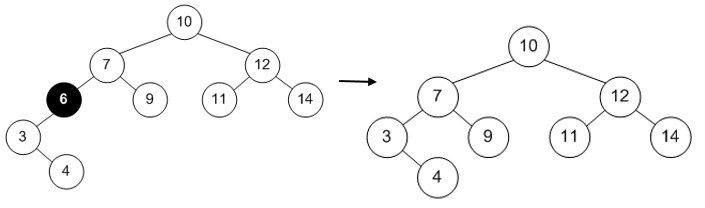
2) у узла имеется наследник, притом только один;

3) у узла есть оба наследника.

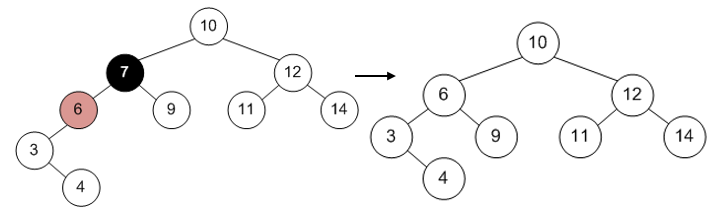
Если у узла нет наследника, то необходимо просто удалить этот узел, а у его родителя обнулить указатель на него. Это самая простая ситуация.



Все слегка усложняется, когда у удаляемого узла имеются наследники.



Если же у удаляемого два наследника, то узел не удаляется, а его значение заменяется на максимум левого поддерева. После этого удаляется максимум левого поддерева.

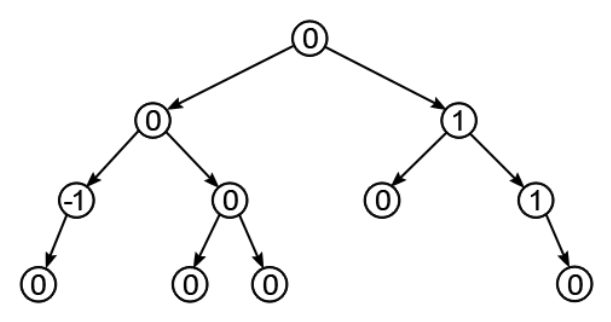


Максимум левого поддерева имеет не более одного наследника, так что он удаляется просто, аналогично ситуации, рассмотренной выше.Функция вставки аналогична функции поиска: необходимо пройти по дереву и вставить узел в нужное место. Для этого следует выбрать левое или правое поддерево корневого узла, а затем рекурсивно перемещаться по выбранному поддереву до тех пор, пока не будет найдена позиция для вставки узла.

*АВЛ Дерево*

Сбалансированным называется такое двоичное дерево поиска, в котором высота каждого из поддеревьев, имеющих общий корень, отличается не более чем на некоторую константу, и при этом выполняются условия, характерные для двоичного дерева.

АВЛ-дерево – сбалансированное двоичное дерево поиска с константой отличия, равной 1. Для узлов АВЛ-дерева определен коэффициент сбалансированности. Это разность высот правого и левого поддеревьев, принимающая одно значения из множества {–1, 0, 1}. Вместо привычных значений узла отображены соответствующие коэффициенты сбалансированности.



Отличительной особенностью сбалансированного дерева является эффективность в обработке. Действительно, ведь максимальное количество шагов, которое необходимо для обнаружения нужного узла, равно количеству уровней самого бинарного дерева поиска.

Так как поддеревья практически симметричны, высота дерева сводится к оптимальному минимуму. Это позволяет ускорить поиск, что сказывается на общей производительности.

Для АВЛ-деревьев определены следующие основные операции:

1) добавление узла в дерево;

2) поиск по дереву;

3) удаление узла;

4) балансировка дерева.

В процессе обработки АВЛ-дерева балансировка может нарушиться, тогда потребуется осуществить операцию балансировки. Если после выполнения операций вставки или удаления коэффициент сбалансированности какого-либо узла АВЛ-дерева становиться равен 2, то необходимо выполнить операцию балансировки.

Операция балансировки осуществляется путем поворота узлов (изменения связей в поддереве). Вращения не меняют свойств бинарного дерева поиска. Различают несколько типов вращения:

1) малый правый поворот;

2) большой правый поворот;

3) малый левый поворот;

4) большой левый поворот.

Оба типа больших поворотов являются комбинацией малых вращений (право-левым или лево-правым поворотом).

Операция вставки нового узла в АВЛ-дерево выполняется рекурсивно. По ключу данного узла производится поиск места вставки: спускаясь по дереву вниз, алгоритм сравнивает ключ добавляемого узла со встречающимися ключами, далее происходит вставка нового элемента.

По возвращению из рекурсии выполняется балансировка. Для осуществления балансировки следует знать, с каким из рассмотренных ранее случаев дисбаланса имеется дело.

Именно поэтому процесс вставки узла в АВЛ-дерево отличается от добавления узла в обычное бинарное дерево. Общий алгоритм выглядит следующим образом:

1) поиск места, где должен будет находиться новый узел;

2) добавление узла в дерево на найденную позицию;

3) перерасчет коэффициентов сбалансированности для узлов, находящихся выше добавленного узла;

4) если дерево стало несбалансированным, то необходимо выполнить балансировку.

Удаление узла из АВЛ-дерева сложнее аналогичной операции для простого двоичного дерева и включает в себя следующие этапы:

1) поиск узла, который требуется удалить (в процессе поиска запоминаются пройденные узлы для выполнения последующей балансировки);

2) удаление искомого узла и обновление списка пройденных узлов;

3) выполнение балансировки и перерасчет коэффициентов сбалансированности.

**Постановка задачи.**

Необходимо реализовать программу, которая выполняет следующие действия.

Для БП деревьев:

Необходимо реализовать программу, которая выполняет следующие действия.

1. Формирование бинарного дерева из N элементов:

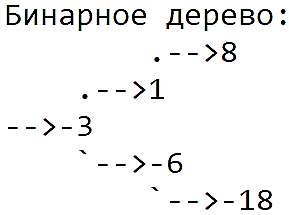
a) пользователь вводит количество элементов N бинарного дерева, которое автоматически заполняется случайными числами (–99 до 99);

б) пользователь вводит в консоль элементы массива, N определяется автоматически по количеству введенных элементов;

в) элементы считываются с файла, в котором хранится массив из чисел, N – количество элементов в файле.

Определение скорости формирования бинарного дерева.

2. Вывод в консоль и файл tree бинарного дерева. Бинарное дерево должно иметь подобный вид:



3. Определение скорости вставки, удаления и получения элемента дерева. В отчете сравните скорость работы бинарного дерева с линейной структурой (двусвязным списком или динамическим массивом) и сделайте выводы.

4. Прямой обход, обратный обход и обход в ширину бинарного дерева.

5. Генерация заданий к практической работе по бинарным деревьям. Необходимо сгенерировать задания в файл output\_task в количестве вариантов, которые введет пользователь. В файл output\_key необходимо вывести короткие ответы к заданиям. В файл output\_ans необходимо вывести развернутые ответы к заданиям. Должны быть представлены задания на создание бинарного дерева поиска из массива значений, удаление и вставка элементов в бинарное дерево поиска.

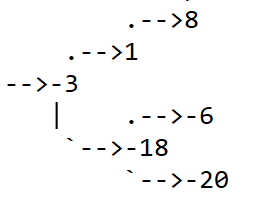
1. Формирование АВЛ-дерева из N элементов:

a) пользователь вводит количество элементов N АВЛ-дерева, которое автоматически заполняется случайными числами (–99 до 99);

б) пользователь вводит в консоль элементы массива, N определяется автоматически по количеству введенных элементов;

Определение скорости формирования АВЛ-дерева.

2. Вывод в консоль AVL\_tree АВЛ-дерева.



3. Определение скорости вставки, удаления и получения элемента дерева.

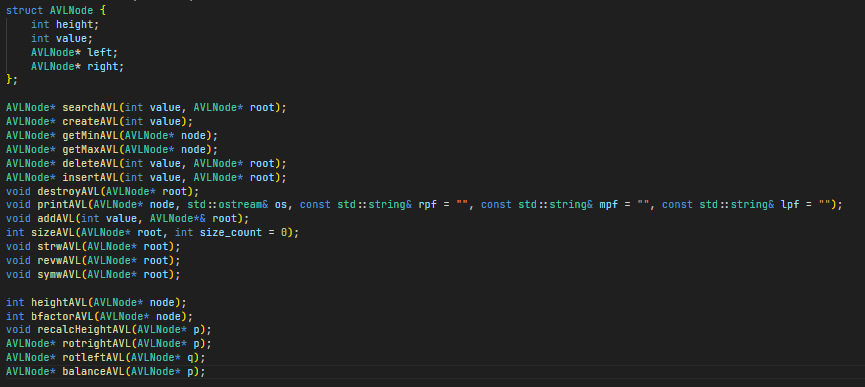
4. Определение скорости проверки на сбалансированность.

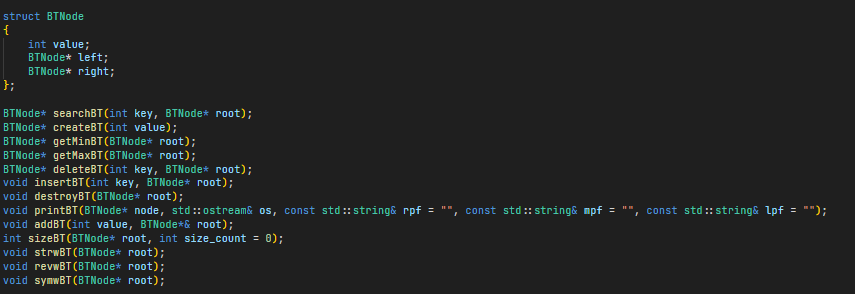
5. Генерация заданий к практической работе по АВЛ-деревьям. Необходимо сгенерировать задания в файл output\_task в количестве вариантов, которые введет пользователь. В файл output\_key необходимо вывести короткие ответы к заданиям. В файл output\_ans необходимо вывести развернутые ответы к заданиям. Должны быть представлены задания на создание АВЛ-дерева из массива значений, удаление и вставка элементов в АВЛ-дерево.

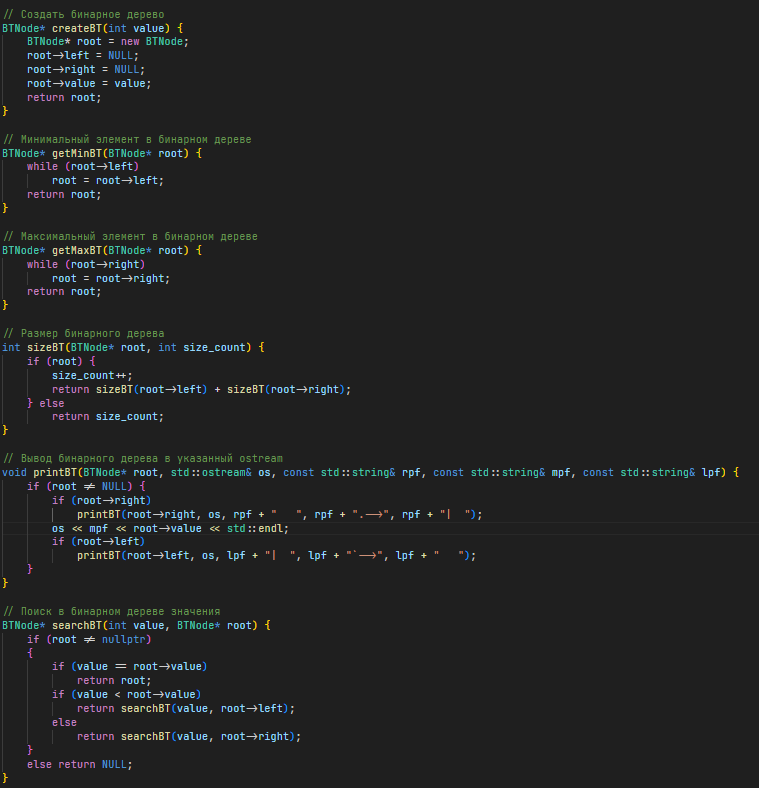
Необходимо добавить в отчёт информацию о проделанной работе в достаточном объеме (в том числе добавить теоретический материал про работу с АВЛ-деревьями). Код необходимо прикрепить к приложению В.

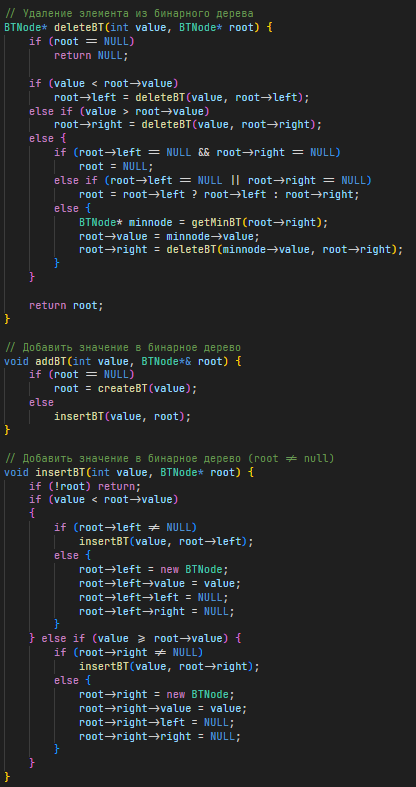
**Выполнение работы.**

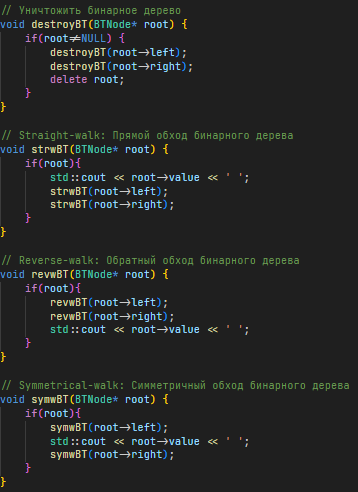
Структура БП и АВЛ деревьев и их функции:

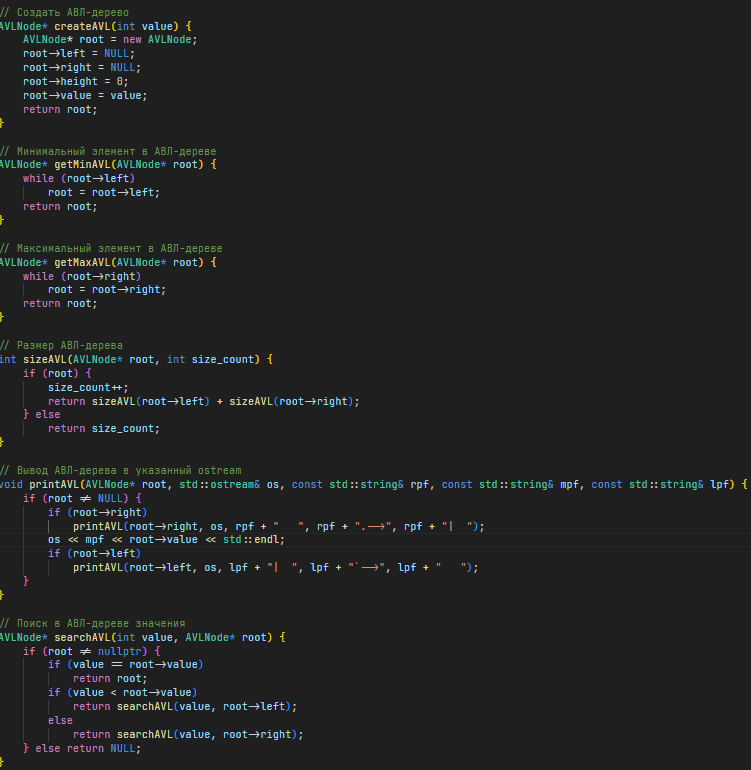


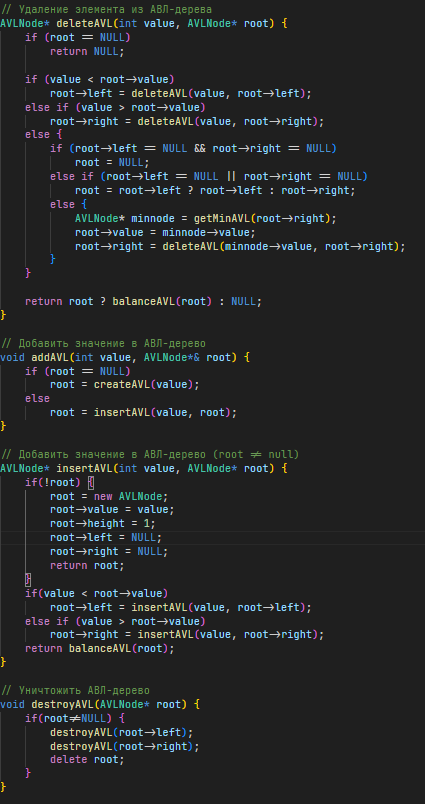


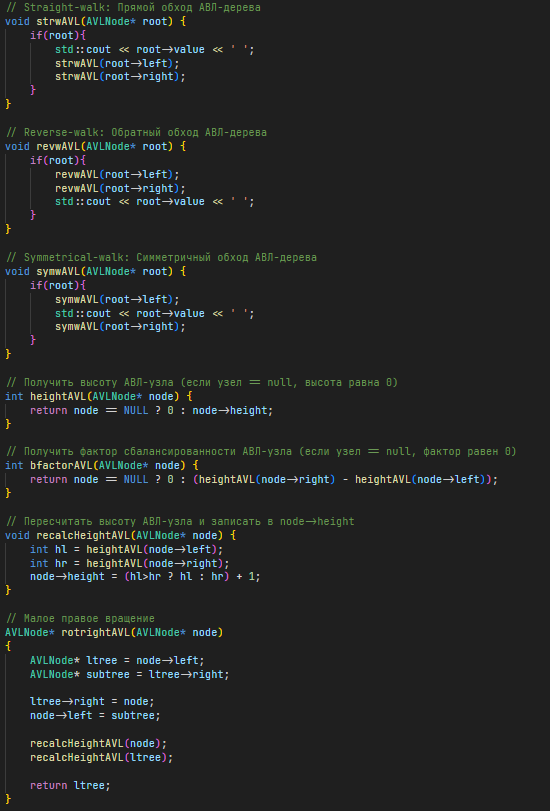


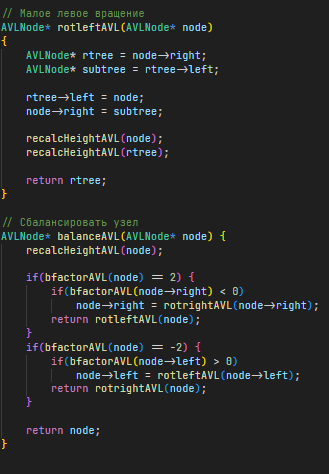




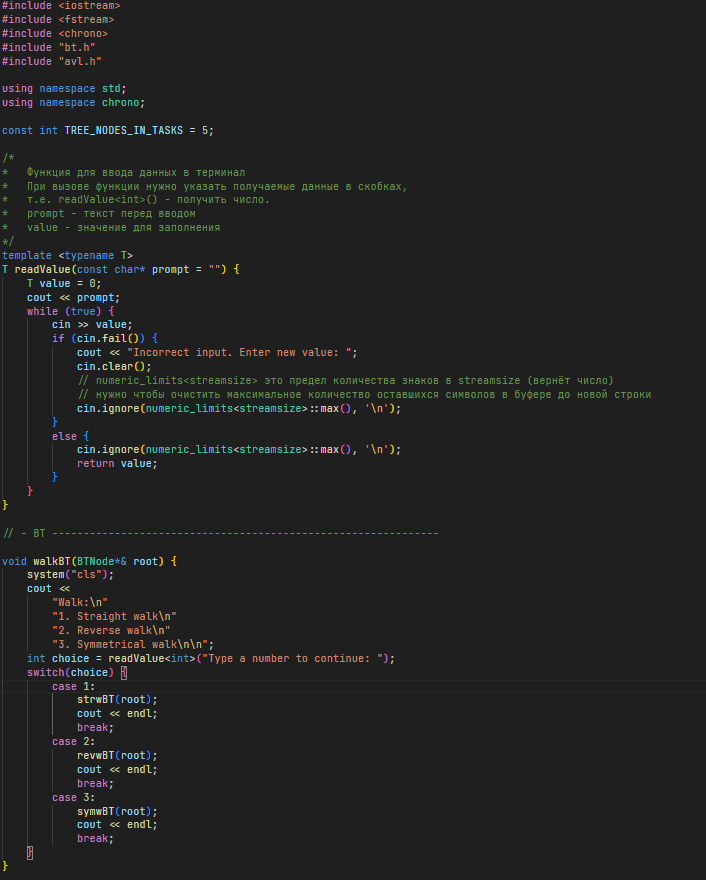


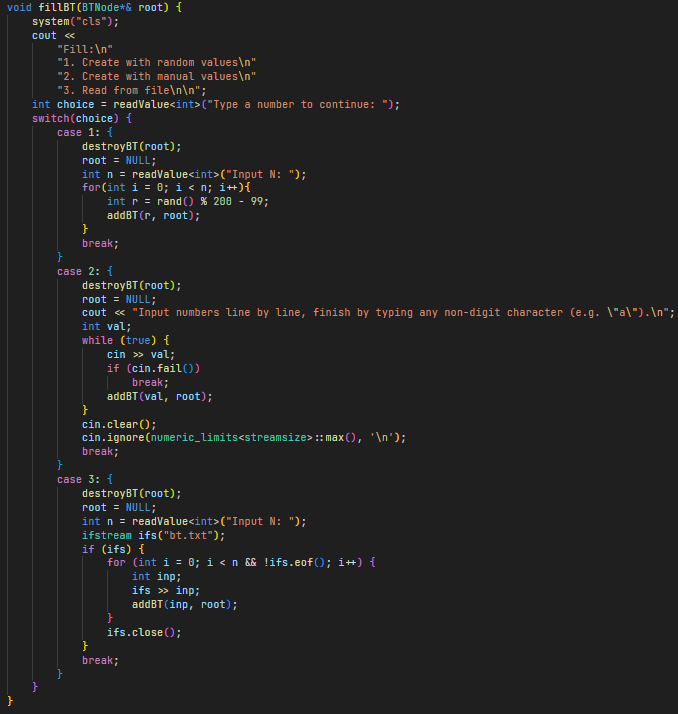




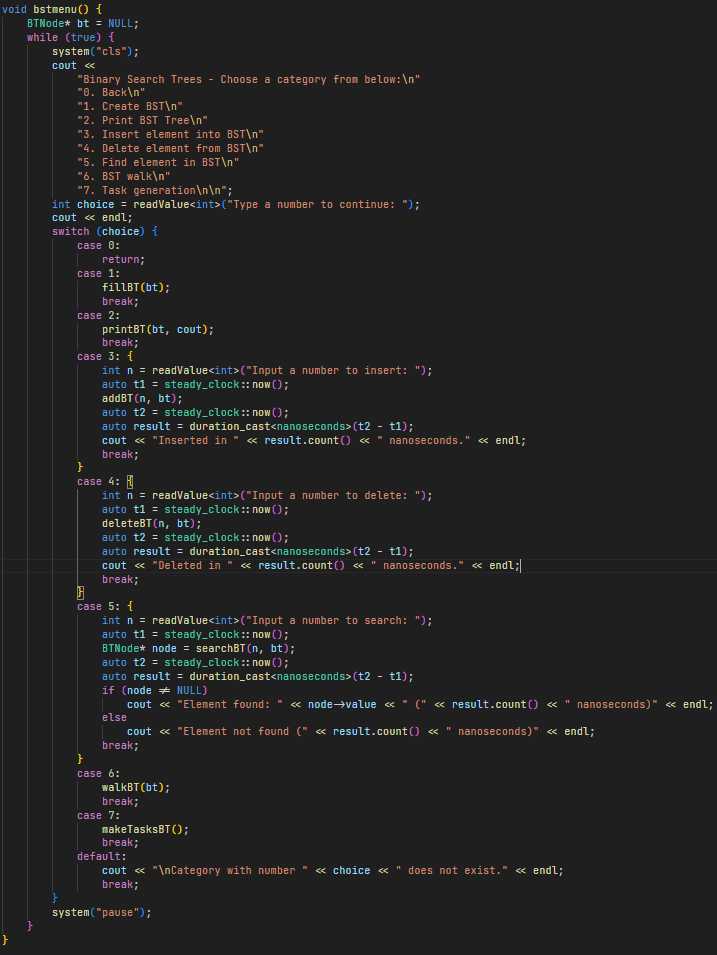


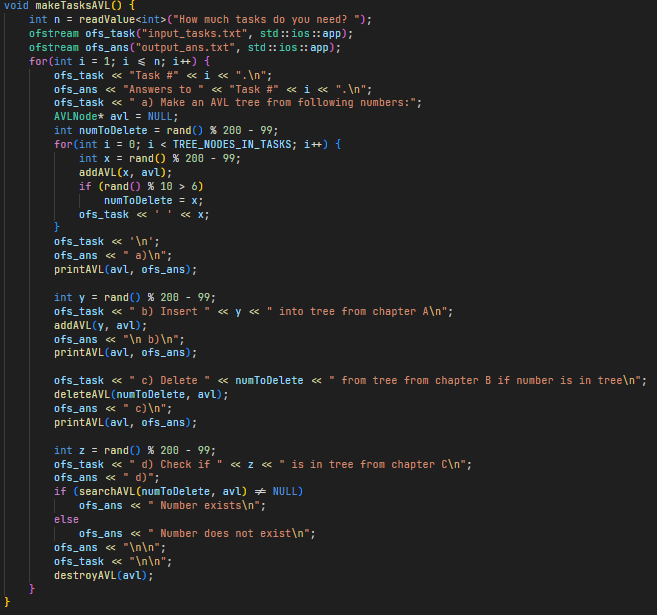
Главный файл программы:

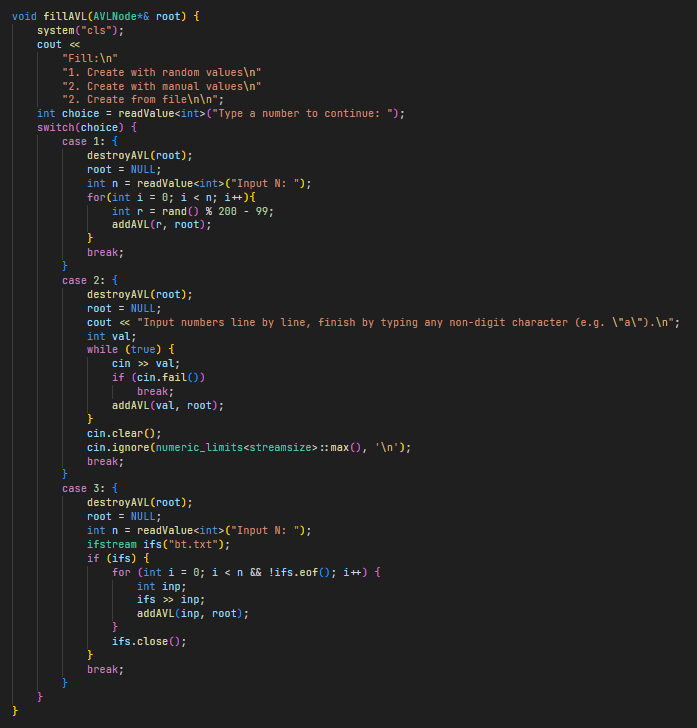




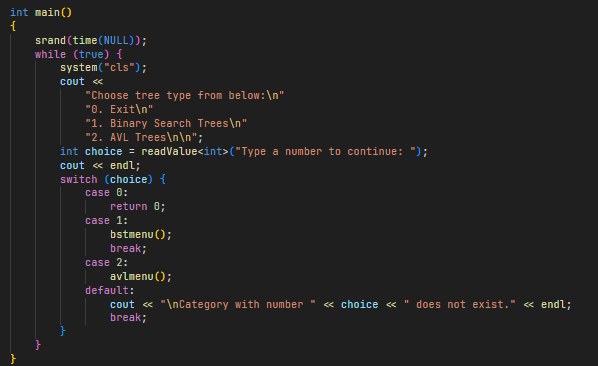












**Выводы.**

Я научился работать с БП и АВЛ деревьями, изучил их строение.