

Требования к программам

1. Программа работает с бинарным деревом объектов типа `student`:

```
class student
{
private:
    char * name;
    int    value;
public:
    ...
};
class tree_node : public student
{
private:
    tree_node * left;
    tree_node * right;
public:
    ...
    friend class tree;
};
class tree
{
private:
    tree_node * root;
public:
    ...
    int read (const char * filename);
    void print (int r);
};
```

Все функции в задании являются членами класса "дерево".

2. Программа должна получать все параметры в качестве аргументов командной строки. Аргументы командной строки:

- 1) r – количество выводимых уровней в дереве,
- 2) `filename` – имя файла, откуда надо прочитать дерево,
- 3) k – параметр задачи.

Например, запуск

```
./a.out 4 a.txt
```

означает, что дерево надо прочитать из файла `a.txt`, выводить не более 4-х уровней дерева.

3. Класс "дерево" должен содержать функцию ввода дерева из указанного файла.
4. Ввод дерева из файла. В указанном файле находится дерево в формате:

```
Слово-1  Целое-число-1
Слово-2  Целое-число-2
...      ...
Слово-n  Целое-число-n
```

где слово – последовательность алфавитно-цифровых символов без пробелов. Длина слова неизвестна, память под него выделяется динамически. При заполнении первый объект типа `student` попадает в корень дерева, каждый новый объект типа `student` добавляется в левое поддерево, если он меньше текущего узла относительно операции сравнения `tree_node::operator<`, и в правое поддерево иначе. **Никакие другие функции, кроме функции ввода дерева, не используют упорядоченность дерева.** Концом ввода считается конец файла. Программа должна выводить сообщение об ошибке, если указанный файл не может быть прочитан или содержит данные неверного формата.

- Решение задачи должно быть оформлено в виде подпрограммы, находящейся в отдельном файле. Получать в этой подпрограмме дополнительную информацию извне через глобальные переменные, включаемые файлы и т.п. запрещается.
- Вывод результата работы функции в функции `main` должен производиться по формату:

```
printf ("Task = %d Result = %d Elapsed = %.2f\n", task, res, t);
```

где

- `task` – номер задачи (1–5),
- `res` – результат работы функции, реализующей решение этой задачи,
- `t` – время работы функции, реализующей решение этой задачи.

Вывод должен производиться в точности в таком формате, чтобы можно было автоматизировать обработку запуска многих тестов.

- Класс "дерево" должен содержать подпрограмму вывода на экран не более чем r уровней дерева. Эта подпрограмма используется для вывода исходного дерева после его инициализации, а также для вывода на экран результата. Подпрограмма выводит на экран не более, чем r уровней дерева, где r – параметр этой подпрограммы (аргумент командной строки). Каждый элемент дерева должен печататься на новой строке и так, чтобы структура дерева была понятна.
- Программа должна выводить на экран время, затраченное на решение.
- Поскольку дерево не изменяется функциями из задач, то для всех задач надо сделать **одну функцию** `main`, в которой прочитать дерево, вывести указанное число уровней на экран, и вызвать все функции задач, выводя результаты и время их работы. Другими словами, после компиляции должен получиться один исполняемый файл `a.out`, а не несколько.

Задачи

- Написать подпрограмму – член класса "дерево", получающую в качестве аргумента целое число k , и возвращающую целое значение, равное количеству поддеревьев, имеющих не более k вершин.
- Написать подпрограмму – член класса "дерево", получающую в качестве аргумента целое число k , и возвращающую целое значение, равное количеству элементов в поддеревьях, имеющих не более k вершин.
- Написать подпрограмму – член класса "дерево", получающую в качестве аргумента целое число k , и возвращающую целое значение, равное количеству поддеревьев, имеющих не более k уровней.
- Написать подпрограмму – член класса "дерево", получающую в качестве аргумента целое число k , и возвращающую целое значение, равное количеству поддеревьев, имеющих не более k элементов в любом уровне.
- Написать подпрограмму – член класса "дерево", получающую в качестве аргумента целое число k , и возвращающую целое значение, равное количеству элементов в его k -м уровне.