Отчет

по лабораторной работе «1067. Disk Tree»

по дисциплине «**Алгоритмы и Структуры данных**»

Авторы:

Полит Алексей Денисович

Факультет: СУиР

Группа: R3235

Преподаватель:Тропченко Андрей Александрович

Санкт-Петербург, 2021г

1. Задачи

Hacker Bill has accidentally lost all the information from his workstation's hard drive and he has no backup copies of its contents. He does not regret for the loss of the files themselves, but for the very nice and convenient directory structure that he had created and cherished during years of work.

Fortunately, Bill has several copies of directory listings from his hard drive. Using those listings he was able to recover full paths (like "WINNT\SYSTEM32\CERTSRV\CERTCO~1\X86") for some directories. He put all of them in a file by writing each path he has found on a separate line.

Your task is to write a program that will help Bill to restore his state of the art directory structure by providing nicely formatted directory tree.

2. Исходные данные

The first line of the input contains single integer number N ($1 \le N \le 500$) that denotes a total number of distinct directory paths. Then N lines with directory paths follow. Each directory path occupies a single line and does not contain any spaces, including leading or trailing ones. No path exceeds 80 characters. Each path is listed once and consists of a number of directory names separated by a back slash ("\").

Each directory name consists of 1 to 8 uppercase letters, numbers, or the special characters from the following list: exclamation mark, number sign, dollar sign, percent sign, ampersand, apostrophe, opening and closing parenthesis, hyphen sign, commercial at, circumflex accent, underscore, grave accent, opening and closing curly bracket, and tilde ("`!#$%&'()-@^_`{}~`").

3. Текст программы

```
 #include <iostream>
#include <string>
#include <sstream>
#include <map>
using namespace std;

class Dir {

        private: map<string, Dir*> childDirs;
```

```
        public:
                Dir() {}

                Dir* getDir(string name) {
                        if (childDirs.find(name) != childDirs.end()) return
childDirs[name];
                        else return createDir(name);
                }

                Dir* createDir(string name) {
                        childDirs[name] = new Dir();
                        return childDirs[name];
                }

                void printTree(string separator = "") {
                        string tabs = " ";
                        tabs += separator;
                        map<string, Dir*> contents(childDirs.begin(), childDirs.end());
                        for (auto it = contents.begin(); it != contents.end(); it++) {
                                cout << separator << it->first << endl;
                                it->second->printTree(tabs);
                        }
                }
};

int main() {
        int n;
        cin >> n;
        Dir* root = new Dir();
        for (int i = 0; i < n; i++) {
                Dir* currentDir = root;
                string fullPath, name;
                cin >> fullPath;
                stringstream ss(fullPath);
                while (getline(ss, name, '\\'))
                        currentDir = currentDir->getDir(name); //здесь '\\' - это
экраннированный символ '\'
        }
        root->printTree();
        return 0;
}
```

9277057	14:17:34 22 мар 2021	Aleksey	1067. Disk Tree	G++ 9.2 x64	Accepted		0.078	3 332 КБ

4. Описание программы

Строим и выводим дерево каталогов. Самы простой вариант - все строки в массив и парсер по первому символу '\', создавая новую цепочку в связанном списке. Это тяжелый и долгий способ.

Нам будет удобо использовать Map, так как файловая система представляет деревовидную структуру. Ключом в данной структуре выступит имя каталога, а значением вложенные катологи.
Dir - инкапсулируем методы создания, получения, вывода директорий. Получаем из fullpath - name, которая предоставляет полный путь к каталогу и часть имени, до первого '\'. Дальше проверим, существует ли в данном каталоге директория name. Нет - создаем. Да - делаем её текущим каталогом и получаем fullpath до '\'. Когда дерево будет полностью заполнено, мы возвращаемся в корень и заново повторяем данные действия.
После всех повторений, мы выводим полученную структуру. Будем рекурсивно опускаться вглубь каждой директории, наращивая на каждом уровне количество пробельных отступов