**Файлова система  
документация**

Имплементацията на файлова система се разбива на няколко подзадачи :

1. Създаване на йерархия от класове представящи различните видове файлове във файловата система.
2. Създаване на дървовидна структура която да представлява системата и да може бързо да намира елементи в нея.
3. Създаване на програма която да обработва командите на потребителя и да следи за невалидни параметри.

Конкретната йерархия е имплементирана чрез един интерфейс (AbstractFile) и три негови наследника – Directory, TextFile и SymLink, както и клас FileSystem който съдържа началото на дървото, построено от директории. Класът CommandPrompt представлява конзолата която обработва потребителския вход и извършва операциите спрямо него. В решаване на задачите които всеки клас извършва се използват предимно стандартните STL класове.

**КЛАСОВА ЙЕРАРХИЯ**

**Class AbstractFile**

Това е базовият клас който се наследява от всички видове файлове във системата. Той съдържа метаданните и извършва някои общи функции, за тяхната обработка от другите класове. Класът е виртуален и съдържа две функции **printMeta()** и **deleteFile()** които всеки наследник трябва да имплементира.

**ЧЛЕН ДАННИ  
string FSysPath**  - пътят на файла във системата. **int FSysNumber –** поредния номер на файла в системата. **int sizeInBytes -**  големината на файла в байтове. **string lastAccessed –**последния достъп до файла. **string lastModified –** последната промяна на файла. **string metaLastModified –** последната промяна на мета данните на файла. **FILE\_TYPE fileType –** типът на файла. **string filename –** името на файла.

**КОНСТРУКТОРИ**Абстрактният файл може да се създаде по подразбиране, в който случай ще е нулевият файл, или с параметри (по един за всяка член данна, в горния ред).

**МЕТОДИ**Класът има селектор за всяка една от член данните.   
**string getTime()** връща текущото време, използва се когато се ъпдейтват метаданните за последна промяна, достъп и т.н.  
**void updateMetaLastModified() –** променя последната промяна по метаданните с текущото време. **void updateLastModified() –** променя последната промяна на файла с текущото време. **void updateLastAccess() –** променя последния достъп до файла с текущото време. **void inheritSize(int) –** наследява промени по големината на файла от някой наследник на файла. **bool isType(FILE\_TYPE) –** показва дали типът на файла е същия с този подаден в параметрите. **FILE\_TYPE tellType() –** връща типа на файла **string fileTypeToString(FILE\_TYPE) –** връща типа на файло под варианта на низ, използва се при принтиране на метаданните.

**ВИРТУАЛНИ ФУНКЦИИ  
virtual void printMeta() = 0  
virtual void deleteFile() = 0**

**Class TextFile**Клът имплементира текстов файл във файловата система. Файлът съдържа своите данни и списък от линкове които сочат към него. При изтриване на файл линковете ще сочат към NULL.

**ЧЛЕН ДАННИ  
string data –**данните които съхранява файла. **list<AbstractFile\*> links –**списък от връзките към файла.

**КОНСТРУКТОРИ**Класът има два конструктора – с параметри и копи конструктор. За конструктора по подразбиране са нужни пътя на директорията съдържаща файла, пореден номер, данните и името на файла.

**МЕТОДИ  
string getData() –** връща низа от данни във файла. **void addLink(AbstractFile\*) –** добавя връзка сочеща към файла. **void concatData(string) –** конкатенира данните на файла с тези подадени в параметрите на функцията. **void printMeta() –** overload на виртуалната функция от абстрактния файл, принтира всички метаданни за файла на стандартния изход. **void deleteFile() –** overload на виртуалната функция от абстрактния файл, занулява всички връзки към файла, директорията която сочи към него трябва да се погрижи за освобождаването на паметта която заема обекта.

**Class SymLink**Класът имплементира връзка към текстов файл във файловата система. Той съдържа само указател към самия файл. При операции над файловете директорията трябва да връща файла към който сочи връзката.

**ЧЛЕН ДАННИ  
TextFile\* file**

**КОНСТРУКТОР**Класът има един конструктор с параметри – указател към файла към който трябва да сочи връзката, пореден номер във файловата система и

**МЕТОДИ  
TextFile\* getLinkedFile() -** връща указател към файла към който сочи връзката. **void printMeta() –** overload на виртуалната функция от абстрактния файл, принтира всички метаданни за файла на стандартния изход. **void deleteFile() –** overload на виртуалната функция от абстрактния файл, занулява връзкata към файла към който е сочил, директорията която сочи към него трябва да се погрижи за освобождаването на паметта която заема обекта.

**Class Directory**Класът имплементира директория във файловата система. Всяка директория (освен корена) съдържа в себе си родителската директория и списък към всичките си наследници. Кореновата директория съдържа само списъка и указателя към родителската си директория е NULL. Тя се пази във класа FyleSystem и се изтрива само в деструктора на този клас.

**ЧЛЕН ДАННИ  
Directory\* parent  
list<AbstractFile\*> children**

**КОНСТРУКТОРИ**Класът има един конструктор с параметри почти същите като при текстовия файл – пътя на родителската директория, пореден номер във файловата система, име на директорията и указател към родителската директория.

**МЕТОДИ  
Directory\* getParent() –** връща указател към родителската директория. **void addChild(AbstractFile\*) –** добавя файл към списъка с наследници, наследниците трябва да бъдат клас наследник на **AbstractFile** – **Directory**, **TextFile** или **SymLink.  
void deleteFIle() –** изтрива файла и всички негови наследници като извиква същата функция при наследника. Така ако директорията съдържа други директории те също ще изтрият своите наследници. **void removeFile(TextFile\*) –** премахва текстов файл от списъка от наследници и почиства паметта която е заема. **void removeLink(SymLink\*) –** премахва връзка към файл от списъка с наследници и почиства паметта която заема. **Directory\* findDirByName(string) –** намира директория по има измежду наследниците, връща **NULL** ако не е намерена такава. **Directory\* findDirRelatiive(string) –** намира директория по път относителен спрямо пътя на директорията, връща **NULL** ако не е намерена такава. **TextFile\* findFileByName(string) –** намира файл по име измежду наследниците, връща **NULL** ако не е намерен такъв. **TextFile\* findFileRelativе(string) –** намира файл по път относителен спрямо пътя на директорията, връща **NULL** ако не е намерен такъв.  **bool nameAvailable(string) –** показва дали името подадено като параметър съществува измежду списъка на наследници на директорията. **void printContent() –** извежда на стандартния изход имената на наследниците (файлове, директории или връзки към файлове). **void printMeta() –** имплементира виртуалната функция наследена от **AbstractFile**, извежда на стандартния изход всички метаданни на файла. **void inheritSize(int) –** наследява промяна в големината на файл и го предава на родителската директория ако има такава; извиква същата функция във **AbstractFile** частта.

**Class FileSystem**Класът имплементира файлова система. Съдържа указател към кореновата директория, нулевия файл и пореден номер. При създаване на файл във файловата система му се дава пореден номер, който вътрешно след това се обновява. Всички файлове във файловата система са динамично заделени и в деструктора се извиква методът **deleteFile()** за да се изтрият всички създадени файлове. Кореновата директория на файловата система има пореден номер 1, а нулевият файл има пореден номер 0, всички други файлове получават номер в ред на създаването.

**ЧЛЕН ДАННИ  
Directory\* root –** указател към кореновата директория; **Directory NullFile –**нулевият файл. **int nextNumber –** текущия пореден номер който се раздава на файлове.

**КОНСТРУКТОРИ**Класът има само един конструктор по подразбиране който създава кореновата директория и нулевия файл.

**МЕТОДИ  
Directory\* getRoot() –** връща кореновата директория. **Directory\* getDir(string) –** намира директория по пълен път във файловата система, връща **NULL**, ако не съществува такава. **TextFile\* getFile(string) –** намира файл по пълен път във файловата система, връща **NULL**, ако не съществува таква. **int getNextNumber() –** връща текущия пореден номер.

**Class CommandPrompt**Класът имплементира конзола за работа с файлова система. Съдържа самата файлова система и указател към текущата директория която се обработва. При обработка на инструкции класът следи за грешки при подаване на данните, но могат да възникнат такива при неправилно подадени параметри или инструкции.

**ЧЛЕН ДАННИ  
FileSystem fs –** файловата система.  
**Directory\* currentDir –** указател към текущата директория която се обработва.  
**bool done –** булева стойност която показва, че потребителя иска да спре да излезе от програмата. Става true когато потребителя напише quit в конзолата.

**КОНСТРУКТОР**Конструкторът е единствен по подразбиране и инициализира файловата система.

**МЕТОДИ  
void work() –** главният цикъл на програмата, тук се извличат инструкциите и параметрите ред по ред и се подават на съответната функция. **void executeOP(string) –** служи като голям switch за операциите, също така разделя операцията от параметрите.

**PRIVATE МЕТОДИ**-Операции   
**void printCurrentPath(string) –** принтира пълният път на текущата директория. **void setCurrentPath(string) -**  намира директория по даден път и променя текущата директория (ако търсената съществува). **void getDirectoryContent(string) –** решава пътя, намира директорията и извежда съдържанието ѝ, ако съществува такава директория. **void concatenateFiles(string) –** конкатенира, създава и извежда файлове. Първо намира файловете които трябва да се конкатенират, след това файла в който трябва да се запишат и ги предава на помощната функция **saveConcatenatedFiles**. **void copyFileInDirectory(string) –** копира файлове в подадена директория. Намира всички файлове които трябва да се копират, намира директорията където тярбва да се копират и след това ги копира. **void deleteFile(string) –** изтрива всички файлове подадени в параметрите. Първо намира директорията в която се намират, след това търси в нея по името на файл и накрая извиква **removeFile** методът на директорията. **void makeDirectory(string) –** създава всички директории подадени в параметрите. **void deleteDirectory(string) -** изтрива всички директории подадени в параметрите. Ако текущата директория е наследник на някоя от директориите които ще бъдат изтрити, тя се променя на кореновата. Кореновата директория не може да бъде изтрита. **void createLinksInDirectory(string) –** създава линкове към файлове в директория. Тъй като директорията е последна в списъка, а трябва да бъде намерена първа, функцията използва стек. Всички други функции използват опашка при работа със списък от файлове или директории. **void getFileMetadata(string) –**извежда метаданните на всички файлве в параметрите. **-**Помощни функции  
**void saveConcatenatedFile(string,string) –** ако подадените данни и път към изходящ файл са празни връща, ако само данните са празни се въвеждат, ако само пътят е празен се извеждат данните. **Directory\* determinePathAndGetDir(string) –** определя вида на директорията - директен наследник на текущата директория, относителен път спрямо текущата директория или пълен път, намира директорията и я връща, връща **NULL** ако не съществува такава. **TextFile\* determinePathAndGetFile(string) -** определя вида на файла - директен наследник на текущата директория, относителен път спрямо текущата директория или пълен път, намира файла и я връща, връща **NULL** ако не съществува такъв. **queue<string> getPaths(string) –** извлича пътищата разделени с разстояние от string и ги връща като опашка от string. **Directory\* createNestedDirectories(string) –** създава вложени директории (използва се ако трябва да се създаде цял път от директории, а не само една).

**ПОДОБРЕНИЯ**Идеи за подобрения на тази имплементация са :

1. Да се запазва цялата файлова система във файлове. Всеки елемент от системата се запазва в отделен файл с име поредния номер. За да се възстанови файловата система се записва още един двоичен файл. Всеки ред от файла е асоциативен списък по номер на файла последван от номерата на всички негови наследници. Файловата система при инциализация ще иска пътя към този файл и същевременно ще чете файла и ще възстановява структурите в паметта.
2. Разделяне на функцията cat на няколко функции – за въвеждане, за извеждане и за конкатениране. В сегашното си състояние синтаксисът на функцията е твърде натоварен и създава предпоставка за възникване на грешки.
3. Намиране на по-ефективен начин на изтриване на файловете в системата.