TP3 C++ : Apache logs analysis

Edern HAUMONT and Nicolas SIX

B3111

13 décembre 2015

1 Specifications

1.1 General specifications

Our program is designed to deal with apache servers log files.

The input of our program (analog) is a correctly built (cf. detailed specs) log file (.log). The output is the list of the ten most visited URL on the server. This list is printed on the standard output. Some options are available to detail the request. The program is able to generate a .dot file which may be viewed with Softwares like Graphviz.

Our program is made to deal quickly with big log file to be easily updated if the treatment needs of the user evolve. That is, if someone wants to modify the application to add a classification level to the Data structure, it must be easy for him/her. The users should not be a programmer, simply a server administrator.

The program is designed for linux systems. But should work on other platforms.

1.2 Log lines

A log line must fulfill several conditions in order to be accepted by the program. Besides the conditions on each information, there is a general structure to check :

- there is a specific order for the informations in a logline
- each information in seperated from the others with a space. However, the request, the referrer, and browser informations are given between double quotes.

Each information of a log line is the object of a small test

Element	Description	Test	Additional informations
Ipv4 adress	The program does not check its validity	TestIpv4	none
User logname	It must be in one world. If there is none, it is replaced by a dash ("-")	TestPseudo&Logname	none
Authenticated User (Pseudo)	It must be in one world. If there is none, it is replaced by a dash ("-")	TestPseudo&Logname	none
Date, time and GMT	As followed : [DD/-Mon/YYYY :HH :MM :SS XGMT] (X replaced by "+" or "-")	TestDate&hour&GMT	These conditions may be added — date < current date — hour between 00 :00 :00 and 23 :59 :59 — GMT between -12 and +12
Total request	As followed (no constraint of world size) : "REQUEST-TYPE requestedURL requestProtocol"	TestRequestType, TestRequestDestination, TestRequest-Protocol	The only request that must be considered as valid by the program is GET, the other are accepted but not treated
Return code	Between 100 and 520, codes above 400 included are considered as fail codes	TestReturnCode	none
Size of transmission	>= 0. If it is unknown, it is replaced by a dash ("-")	TestDataSize	none
Referrer	Given between double quotes	TestReferrer	none
Browser and browser infos	Given between double quotes	TestBrowser	No particular syntax

Log line template : MyIpAdress Logname Pseudo [DD/Mon/YYYY :HH :MM :SS XGMT] "REQUESTTYPE requestedURL requestProtocol" COD SIZE "referrer" "Browser informations given without a specific order"

Exemple : 192.168.0.0 - - [08/Sep/2012 :11 :16 :02 +0200] "GET /temps/4IF16.html HTTP/1.1" 200 12106 "http ://intranet-if.insa-lyon.fr/temps/4IF15.html" "Mozilla/5.0 (Windows NT 6.1; WOW64; rv :14.0) Gecko/20100101 Firefox/14.0.1"

1.3 Accepted requests

The program's name is "analog". It can be called with the name of a log file and possibly several options. Even if the options do not change the way a log file is read, it changes the kind of output or, at least a certain selection of treated informations.

Here is the template of a call :

```
./analog [options] filename.log
```

Here are three available options :

option	description	test
-e	e stands for exceptions. With this option, the program must exclude of the treatment all documents (in the log requests) that have a picture, css or javascript extension. As it is accepted if the program only exclude most common extensions, we chose to let the user define them in a text file (excludedExtensions.txt)	TestRequestE
-g fileName	g stands for graphic. The program must generate a .dot file, compatible with the software graphviz. In this graph file, the nodes are documents and the arcs are a travelling through a document to another. These arcs correspond to the link between a referrer URL and a destination URL in a log line	TestRequestG
-t hour	t stands for time. hour is a number between 0 and 23. This option is used to select only one hour of data between hour and hout+1. That means, the results in the output and/or those in the dot file are those corresponding to the trafic on the server in this interval of time. It is as if the logfile lines out of this interval didn't exist	TestRequestT
none	If there is no option -g, the program will return the 10 most used resources which fill the options. If there are less than 10 entries, it only prints those that are stored.	TestRequestSimple, TestNoLogfile, TestEmptyLogFile

1.4 Data structure

We want a data structure which stores one information for each referrer-destination-hour combination : the number of hits. It must be fast and easily upgradable, so all informations, including those which are useless must also be stored. The Data structure must be optimized for readings in priority. However, if a future user wants to add an additional sorting level, it must be easy for him/her.

2 Conception and Data Structure

2.1 Data storage

For our Data storage structure, we chose to sort informations by referrer, destination (requestedURL), and hour, in this order.

As there are 24 hours in a day, the sorting by hour is easily implemented by an array of 24 cells. For the two other informations though, the number of possible values depends on the log file. Consequently, we chose to use balanced binary trees as an effective dynamic structure.

You can find at the last page some patterns that resume the Data storage structure at the end of the document.

To resume, we have a main binary tree ordered by destination. Each cell contains other binary trees ordered by referrer. Each cell contains a static array ordered by hour. Each cell contains a vector of other info.

We chose to keep other infos in a class so that, if the user wants to add a level of sorting, he/she just has to take it from this class (as the parser already gets it)

2.2 Classes

We have 3 classes. You can find at the end of the document the general pattern of the application, with the interactions between classes.

2.2.1 DataManager

DataManager is our main class. It contains our Data structures.

It has two main methods :

```
int LoadLogFile(const std::string \&logFilePath);
```

This method is used to load a logfile in the program. The parameter logFilePath is the path to the file.

```
int Request(const bool optionT=false,const int tHour=-1,const bool optionE=false,\
            const bool optionG=false,const std::string \&outputFile="");
```

This method generates the outputs of the program. The parameters correspond to user options, outputFile is the path to a generated .dot file

2.2.2 GraphGenerator

The only usage of this class is its ability to generate a .dot file. This file is created at the construction of the object.

There are two public methods in this class :

```
int addNodeToGraph(const std::string &nodeName);
```

This method adds a Node to the graph, the parameter is the name of the Node.

```
int addLinkToGraph(const std::string &nodeNameFrom, const std::string &nodeNameTo,\
                   const std::string &linkLabel);
```

This method adds a link between the two nodes of the graph given in parameters. The link has a label (3rd parameter).

2.2.3 LogOtherInfos

This class contains all infos of a log line that are not used to store informations in our data structure. As the information is already parsed by the DataManager class, if someone wants to upgrade the application, he/she will only have to take the attribute he/she wants from this class and make it a sorting element.

3 Appendix

FIGURE 1 – data structure

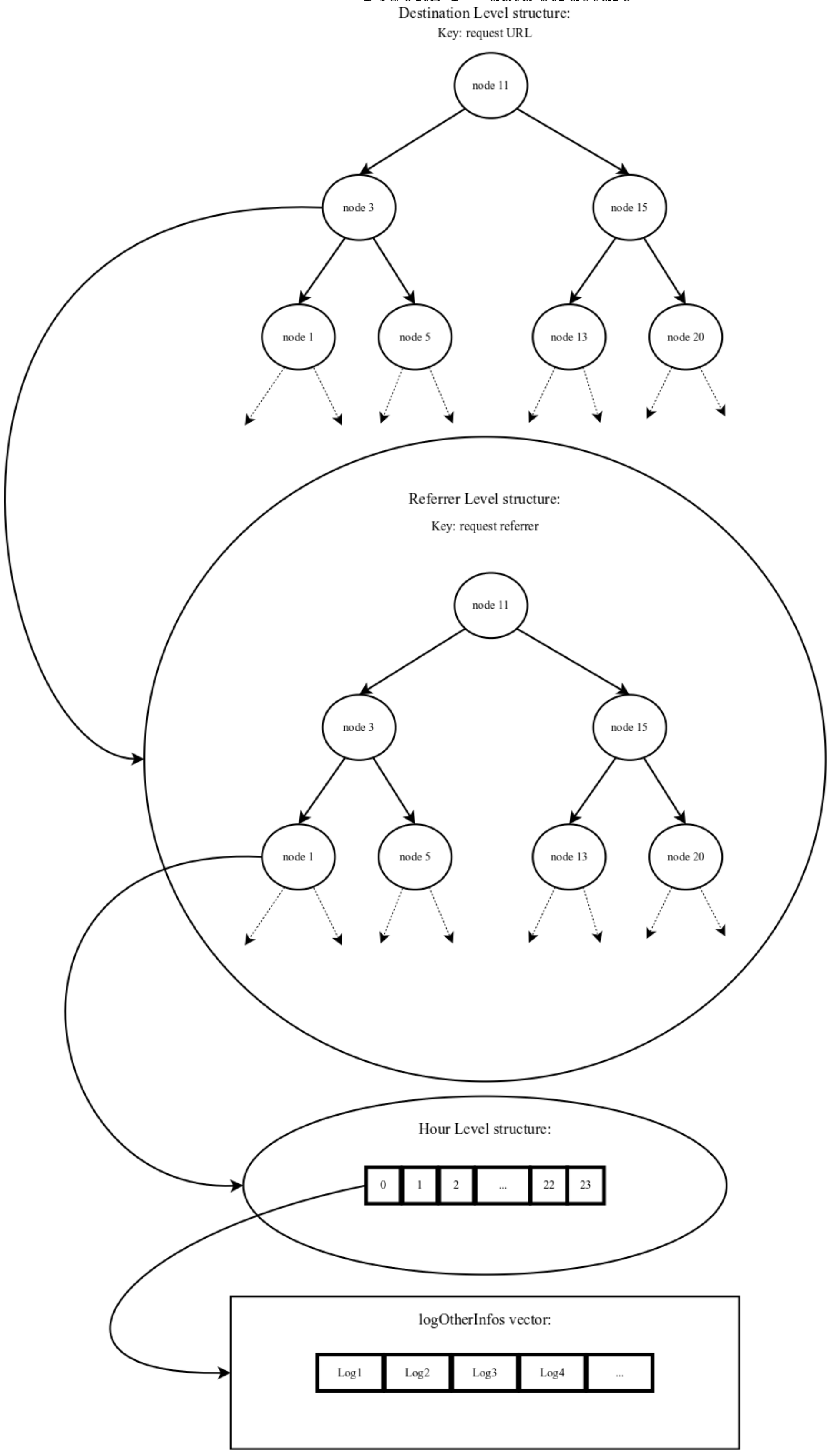
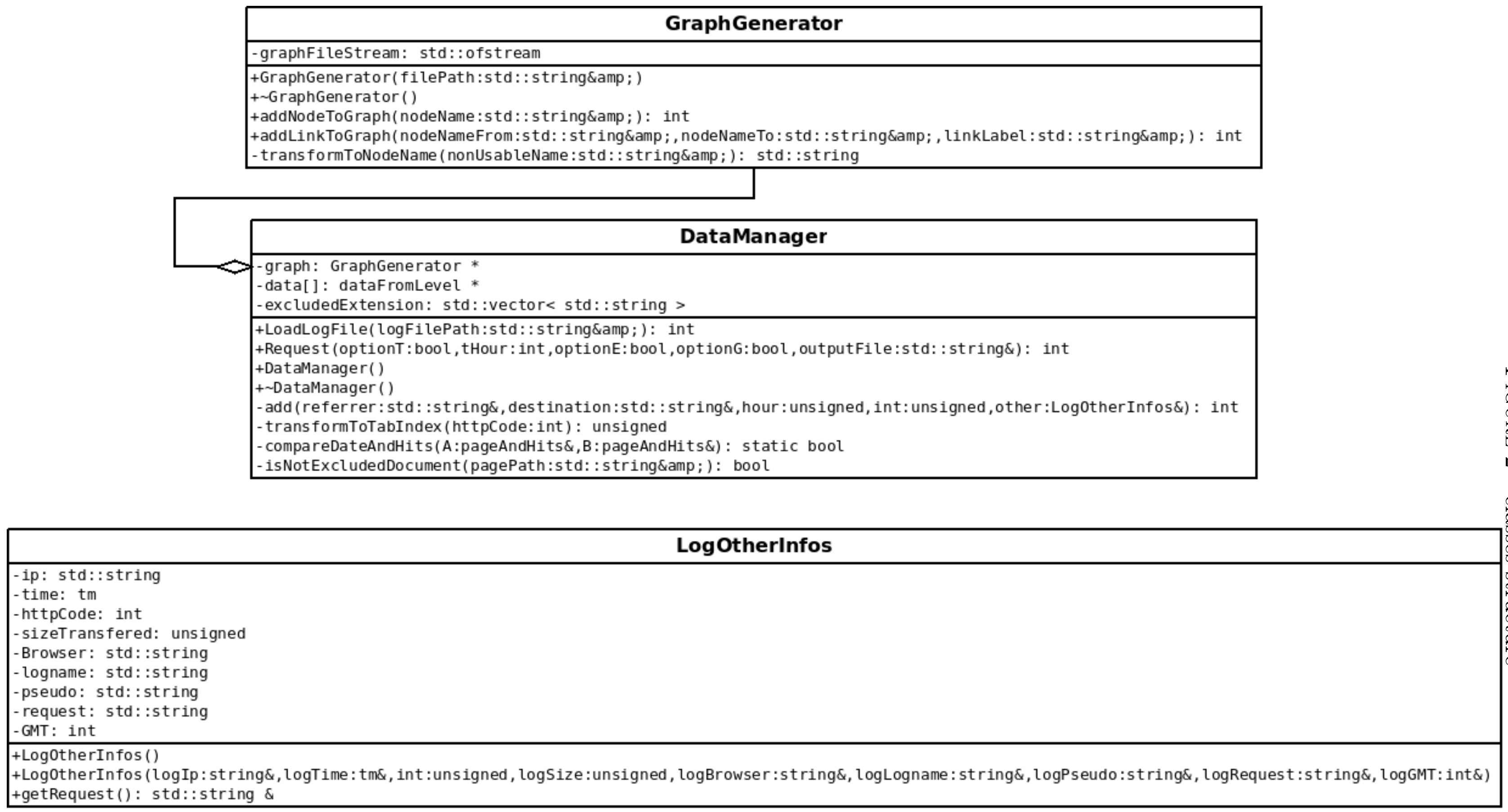


FIGURE 2 – classes structure


GraphGenerator
-graphFileStream: std::ofstream
+GraphGenerator(filePath:std::string&amp;)
+~GraphGenerator()
+addNodeToGraph(nodeName:std::string&amp;): int
+addLinkToGraph(nodeNameFrom:std::string&amp;,nodeNameTo:std::string&amp;,linkLabel:std::string&amp;): int
-transformToNodeName(nonUsableName:std::string&amp;): std::string
DataManager
-graph: GraphGenerator *
-data[]: dataFromLevel *
-excludedExtension: std::vector< std::string >
+LoadLogFile(logFilePath:std::string&amp;): int
+Request(optionT:bool,tHour:int,optionE:bool,optionG:bool,outputFile:std::string&): int
+DataManager()
+~DataManager()
-add(referrer:std::string&,destination:std::string&,hour:unsigned,int:unsigned,other:LogOtherInfos&): int
-transformToTabIndex(httpCode:int): unsigned
-compareDateAndHits(A:pageAndHits&,B:pageAndHits&): static bool
-isNotExcludedDocument(pagePath:std::string&amp;): bool
LogOtherInfos
-ip: std::string
-time: tm
-httpCode: int
-sizeTransfered: unsigned
-Browser: std::string
-logname: std::string
-pseudo: std::string
-request: std::string
-GMT: int
+LogOtherInfos()
+LogOtherInfos(logIp:string&,logTime:tm&,int:unsigned,logSize:unsigned,logBrowser:string&,logLogname:string&,logPseudo:string&,logRequest:string&,logGMT:int&)
+getRequest(): std::string &

```
NAME
    analog, a program that treats apache log files

SYNOPSIS
        analog [OPTION]... <apache-server-logfile.log>

DESCRIPTION
        List  information  about the FILEs (the current directory by default).
        Reads the log file and returns top 10 visited documents in the fol,
        with associated numbers of hits

        Sort entries by the number of hits and then by alphabetical order.

        -e
                excludes images, css and javascript extensions from the results.
                Excluded extensions can be modified in "excludedExtensions.txt"

        -t HH
                limits the results to requests between the hour HH and HH+1.

        -g <graphvizfile.dot>
                exports the apache log analysis in a graphviz file given in
                parameter. Nodes are requested web documents and links repres-
                ent the navigation between the documents.

        A logfile line must be written as follow

        AnIpAdress Logname Pseudo [DD/Mon/YYYY:HH:MM:SS XGMT] "REQUESTTYPE re-
        questedURL requestProtocol" httpreturncode sizetransfered "referrer"
        "Browser informations given without a specific order"

        Exit status:
                0   if OK,

                1   if error when opening the log file,

REPORTING BUGS
        github.com/anliec/TP-Cpp/issues

AUTHOR
        Edern Haumont : edern.haumont@insa-lyon.fr
        Nicolas Six   : nicolas.six@insa-lyon.fr

COPYRIGHT
       Copyright © 2014 Edern Haumont & Nicolas Six.  License  GPLv3+:  GNU
       GPL version 3 or later <http://gnu.org/licenses/gpl.html>.
       This  is  free  software:  you are free to change and redistribute it.
       There is NO WARRANTY, to the extent permitted by law.

Analog TP-Cpp 3                December 2015                        ANALOG(1)
```

```
#makefile

CC=g++
CFLAGS=-std=c++11 -Wall
LDFLAGS=
EXEC=analog
SRC=main.cpp DataManager.cpp LogOtherInfos.cpp GraphGenerator.cpp
OBJ=$(SRC:.cpp=.o)

all: $(EXEC)

analog: $(OBJ)
	$(CC) -o $@ $^ $(LDFLAGS)

%.o: %.cpp
	$(CC) -o $@ -c $< $(CFLAGS)

DataManager.cpp: DataManager.h

LogOtherInfos.cpp: LogOtherInfos.h

GraphGenerator.cpp: GraphGenerator.h

.PHONY:clean

clean:
	rm -rf $(OBJ) $(EXEC)
```

```cpp
#include <iostream>
#include <stdio.h>
#include <stdlib.h>

#include "DataManager.h"

// don't use: "using namespace std;" to keep clear that we use std and
// not any other library and by the same way keeping ready to use an other
// library than the std.

int main(int argc, char * argv[])
{
    DataManager manager;

    bool optionE = false;
    bool optionT = false;
    bool optionG = false;
    int tHour = 0;
    std::string gFilePath = "";
    std::string logFilePath = "";
    int currentArg = 1; //pass the first arg which is the program name (+ path)

    //read the input arguments:

    while (currentArg < argc-1)
    {
        if(argv[currentArg][0] == '-')
        {
           switch (argv[currentArg][1])
           {
               case 'e':
                   optionE = true;
                   break;
               case 't':
                   optionT = true;
                   currentArg++; //if the argument is "-t" then the next must be the hour
                   tHour = atoi(argv[currentArg]);
                   break;
               case 'g':
                   optionG = true;
                   currentArg++; //if the argument is "-g" then the next must be the file path
                   gFilePath = argv[currentArg];
                   break;
           }
        }
        currentArg++;
    }
    if(currentArg >= argc)
    {
        std::cout << "wrong number of arguments" << std::endl;
        return 1;
    }
    logFilePath = argv[argc-1];

    if(manager.LoadLogFile(logFilePath)==1)
    {
         return 1;
    }
    manager.Request(optionT,tHour, optionE, optionG,gFilePath);
    return 0;
}
```

```
/*************************************************************************
                           DataManager  -  description
                             -------------------
    debut                : 23/11/2015
    copyright            : (C) 2015 by Edern Haumont & Nicolas Six
*************************************************************************/


//-------- Interface of the class DataManager (file DataManager.h) -------
#ifndef TP_CPP_DATAMANAGER_H
#define TP_CPP_DATAMANAGER_H

//---------------------------------------------------------------- Includes
#include <iostream>
#include <string>
#include <fstream>
#include <map>
#include <vector>
#include <sstream>

#include "LogOtherInfos.h"
#include "GraphGenerator.h"

//--------------------------------------------------------------- Constants
const int DATA_TAB_SIZE = 2;

//-----------------------------------------------------------------Typedefs
typedef std::vector< LogOtherInfos > dataHourLevel;
typedef std::map< std::string, dataHourLevel* > dataDestinationLevel;
typedef std::map< std::string, dataDestinationLevel* > dataFromLevel;

typedef std::pair< std::string, int> pageAndHits;

//------------------------------------------------------------------------
// The class DataManager is the main class of the program. It contains all
// data structures, creates, manages and make calculations on them.
//------------------------------------------------------------------------
class DataManager {

//------------------------------------------------------------------ PUBLIC
public:


//---------------------------------------------------------- Public methods
    int LoadLogFile(const std::string &logFilePath);
    // User guide :
    // This method is used to load a logfile in the program.
    // The parameter logFilePath is the path to the file.

    int Request(const bool optionT=false,const int tHour=-1,const bool optionE=false,const bool 
optionG=false,const std::string &outputFile="");
    // User guide :
    // Generate the outputs of the program.
    // The parameters correspond to user options, outputFile is the path
    // to a generated .dot file

//-------------------------------------------- Constructors - destructors
    DataManager();
    // Use at the beginning of the program
    virtual ~DataManager();
    // Use at the end of the program

//--------------------------------------------------------- Private methods
private:
    int add(const std::string &referrer,const std::string &destination, unsigned hour, unsigned int 
httpCode,const LogOtherInfos &other);


    unsigned transformToTabIndex(int httpCode) const;

    static bool compareDateAndHits(const pageAndHits &A, const pageAndHits &B);
    bool isNotExcludedDocument(const std::string &pagePath) const;
    /* return: false if the extension is in the list of excluded extension
     */

//------------------------------------------------------- Private atributs

    GraphGenerator * graph;
    dataFromLevel * data[DATA_TAB_SIZE];
    std::vector< std::string > excludedExtension;

};


#endif //TP_CPP_DATAMANAGER_H
```

```
/*************************************************************************
                           DataManager  -  description
                             -------------------
    begin                : 23/11/2015
    copyright            : (C) 2015 by Edern Haumont & Nicolas Six
*************************************************************************/

//------- Realisation of the class DataManager (file DataManager) --------

//---------------------------------------------------------------- INCLUDE

//-------------------------------------------------------- System include
#include <algorithm>
//------------------------------------------------------ Personal include
#include "DataManager.h"
#include "config.h"

// don't use: "using namespace std;" to keep clear that we use std and
// not any other library and by the same way keeping ready to use an other
// library than the std.

//----------------------------------------------------------------- PUBLIC

//-------------------------------------------------------- Public methods

// Constructor
DataManager::DataManager() {
    std::ifstream extensionFile (EXTENSION_FILE);
    for(std::string extension ; std::getline(extensionFile,extension) ; )
    {
        excludedExtension.push_back(extension);
    }
    extensionFile.close();

    for (int i = 0; i < DATA_TAB_SIZE; ++i)
    {
        data[i] = nullptr;
    }
}

// Destructor
DataManager::~DataManager()
// Algorithm : Run through the graph and delete all dynamic elements.
{
    for (int c = 0; c < DATA_TAB_SIZE; c++)
    {
        if(data[c] != nullptr)
        {
            //iterate through the from node:
            for(dataFromLevel::iterator f=data[c]->begin() ; f!=data[c]->end() ; ++f)
            {
                //iterate through the referrer branches:
                for(dataDestinationLevel::iterator d=f->second->begin() ; d!=f->second->end() ; ++d)
                {
                    delete [] d->second;
                }
                delete f->second;
            }
            delete data[c];
        }
    }
}

int DataManager::LoadLogFile(const std::string &logFilePath)
// Algorithm :
// Open a log file. Reads line by line its content until end of file is reached.
// Each line is put in a string stream. Then it is parsed to obtain all its characteristics
{
    std::ifstream logFile(logFilePath, std::ios::in);  // on ouvre le fichier en lecture
    if(!logFile)
    {
        std::cerr << "erreur lors de l'ouverture du fichier de log: " << logFilePath << std::endl;
        return 1;
    }
    else
    {
        std::string logLine;

        std::string ip;
        tm time;
        unsigned int httpCode;
        std::string sizeTransfered;
        unsigned int sizeTransferedValue;
        std::string browser;
        std::string logname;
        std::string pseudo;
        std::string request;
        int GMT;
        std::string unusedBuffer;
        std::string dateBuffer, GMTBuffer;
        std::string protocolRequest;
        std::string URLRequest;
        std::string refferer;

        //loops until end of file or bad reading
        while(getline(logFile,logLine))
        {
            try
            {
                std::stringstream ss(logLine);
                ss >> ip >> logname >> pseudo >> dateBuffer >> GMTBuffer >> request;
                std::string bufferString;
                getline(ss, bufferString, '"');
                unsigned long lastSpace = bufferString.find_last_of(" ");
                URLRequest = bufferString.substr(1,lastSpace-1);
                protocolRequest = bufferString.substr(lastSpace+1, bufferString.length()-lastSpace-1);
                ss >> httpCode >> sizeTransfered >> refferer;
                if(sizeTransfered.compare("-") ==0)
                {
                    sizeTransferedValue = 0;
                }
                else
                {
                    sizeTransferedValue = (unsigned)atoi(sizeTransfered.c_str());
                }
                request.append(" ");
                request.append(URLRequest);
                request.append(" ");
                request.append(protocolRequest);

                //date extraction
                time.tm_mday = atoi(dateBuffer.substr(1,2).c_str());
                std::string Month [] =
{"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"};
                for(int i=0;i<12;i++)
                {
                    if (Month[i].compare(dateBuffer.substr(4, 3)) == 0) {
                        time.tm_mon = i;
                        break;
                    }
                }
                time.tm_year = atoi(dateBuffer.substr(8,4).c_str());
                time.tm_hour = atoi(dateBuffer.substr(13,2).c_str());
                time.tm_min = atoi(dateBuffer.substr(16,2).c_str());
                time.tm_sec = atoi(dateBuffer.substr(19,2).c_str());
                GMT = atoi(GMTBuffer.substr(1,4).c_str()); // /100 ? ( 0200 -> 2h)
                GMT *= (GMTBuffer.substr(0,1) == "-") ? -1 : 1;

                // referrer extraction and management
                if(refferer.length()>32 && refferer.substr(1,32).compare("http://intranet-if.insa-
lyon.fr/")==0)
                {
                    refferer = refferer.substr(32);
                }
```

```cpp
                else
                {
                    refferer = refferer.substr(1);
                }
                refferer = refferer.substr(0,refferer.size()-1);

                getline(ss, unusedBuffer, '"');
                getline(ss, browser, '"');

                LogOtherInfos other
(ip,time,httpCode,sizeTransferedValue,browser,logname,pseudo,request,GMT);
                //add to structure
                add(refferer, URLRequest, (unsigned)time.tm_hour, httpCode, other);
            }
            catch (std::exception e)
            {
                std::cerr << e.what() << " when reading the log file" << std::endl;
            }
        }
    }



    return 0;
} // end of method

int DataManager::Request(const bool optionT, const int tHour, const bool optionE, const bool optionG,
const std::string &outputFile)
// Algorithm : depends on the options.
// Runs through the structure to find most popular URL.
// if optionG checked, associate referrer to destination URL in a .dot
{
    if(optionG)
    {
        graph = new GraphGenerator(outputFile);
    }


    int hourMin=0,hourMax=24;
    if(optionT)
    {
        hourMin = tHour;
        hourMax = tHour+1;
    }

    std::vector< pageAndHits > pageHit;

    for (int c = 0; c < 1; c++)
    {
        if(data[c] != nullptr)
        {
            //iterate through the from node:
            for(dataFromLevel::iterator f=data[c]->begin() ; f!=data[c]->end() ; ++f)
            {
                //option -e filter: if the option is activated then only select the specified extension
                if( !optionE || isNotExcludedDocument(f->first) )
                {
                    int numberOfHitsByPage=0;

                    //iterate through the referrer branches:
                    for(dataDestinationLevel::iterator d=f->second->begin() ; d!=f->second->end() ; +
+d)
                    {
                        int numberOfHitsByReferrer = 0;
                        for (int h=hourMin ; h<hourMax ; h++)
                        {
                            for (unsigned i = 0; i < d->second[h].size(); ++i)
                            {
                                if(d->second[h].at(i).getRequest().substr(1,3).compare("GET")==0)
                                {
                                    numberOfHitsByReferrer++;
                                }
                            }
                        }
                        if(optionG)
                        {
                            graph->addLinkToGraph(f->first,d->first,std::to_string
(numberOfHitsByReferrer));
                        }
                        numberOfHitsByPage += numberOfHitsByReferrer;
                    }
                    if(numberOfHitsByPage != 0)
                    {
                        pageAndHits tuple(f->first, numberOfHitsByPage);
                        pageHit.push_back(tuple);
                    }
                }
            }
        }
    }

    if(optionG)
    {
        delete graph;
    }

    std::sort(pageHit.begin(),pageHit.end(),&compareDateAndHits);

    for (unsigned i=0 ; i<10 && i<pageHit.size() ; i++)
    {
        std::cout << pageHit.at(i).first << " (" << pageHit.at(i).second << " hits)" << std::endl;
    }

    return 0;
}

int DataManager::add(const std::string &referrer, const std::string &destination, unsigned int hour, \
                     unsigned int httpCode, const LogOtherInfos &other)
// Algorithm : runs through the structure
{
    unsigned int indexHttpCode = transformToTabIndex(httpCode);

    dataDestinationLevel* referrerMap;

    if(data[indexHttpCode] == nullptr)
    {
        data[indexHttpCode] = new dataFromLevel();
    }

    //try to add the referrer level to the destination level (if he already exist does nothing)
    if(data[indexHttpCode]->find(destination) == data[indexHttpCode]->end())
    {
        referrerMap = new dataDestinationLevel();
        std::pair<std::string,dataDestinationLevel*> insertionPairDest(destination, referrerMap);
        data[indexHttpCode]->insert(insertionPairDest);
    }
    else
    {
        referrerMap = data[indexHttpCode]->at(destination);
    }

    //try to add the hour level to the referrer level (if he already exist does nothing)
    if(referrerMap->find(referrer) == referrerMap->end())
    {
        dataHourLevel * tempHourLevelVector = new dataHourLevel[24];
        for (int i = 0; i < 24; i++)
        {
            dataHourLevel temp;
            tempHourLevelVector[i] = temp;
        }
        std::pair<std::string,dataHourLevel*> insertionPairHour(referrer, tempHourLevelVector);
        referrerMap->insert(insertionPairHour);
    }
```

```cpp
    dataHourLevel * hourLevel = referrerMap->at(referrer);
    hourLevel[hour].push_back(other);

    return 0;
}




bool DataManager::compareDateAndHits(const pageAndHits &A, const pageAndHits &B)
// function made to order the values by number of hits and then by name of the page
{
    return (A.second > B.second) || ((A.second == B.second) && (A.first.compare(B.first)<0));
}

bool DataManager::isNotExcludedDocument(const std::string &pagePath) const
{
    if(pagePath.find('.') != std::string::npos)
    {
        std::string extension = pagePath.substr( pagePath.find_last_of('.'));

        for (unsigned i = 0; i < excludedExtension.size(); ++i) {
            if(extension.compare(excludedExtension.at(i))==0)
            {
                return false;
            }
        }
    }

    return true;
}

unsigned DataManager::transformToTabIndex(int httpCode) const {
    //equivalent to: (httpCode-100)/300 but handel error case:
    if(httpCode >= 100 && httpCode < 400)
    {
        return 0;
    }
    else // if on [400;600[ or if any error
    {
        return 1;
    }
}
```

```cpp
/*************************************************************************
                     GraphGenerator  -  description
                             -------------------
    debut                : 23/11/2015
    copyright            : (C) 2015 by Edern Haumont & Nicolas Six
*************************************************************************/

#ifndef TP_CPP_GRAPHGENERATOR_H
#define TP_CPP_GRAPHGENERATOR_H

//------------------------------------------------------------------ Includes
#include <iostream>
#include <fstream>

//----------------------------------------------------------------- Constants
const int FILE_ERROR = 80;

//--------------------------------------------------------------------------
// The class GraphGenerator handle all that is related to the creation of
// the graph file
//--------------------------------------------------------------------------
class GraphGenerator
{
//------------------------------------------------------------------- PUBLIC
public:
//----------------------------------------------------------- Public methods
    int addNodeToGraph(const std::string &nodeName);
    // adds a node to the graph. Label given in parameter
    int addLinkToGraph(const std::string &nodeNameFrom, const std::string &nodeNameTo, const
std::string &linkLabel);
    //adds a link between two nodes given in parameters. Label given in parameter
//-------------------------------------------------- Constructors - destructors
    GraphGenerator(const std::string &filePath);
    virtual ~GraphGenerator();

//------------------------------------------------------------------ PRIVATE
private:
//---------------------------------------------------------- Private methods
    std::string transformToNodeName(const std::string &nonUsableName) const;

private:
    std::ofstream graphFileStream;
};


#endif //TP_CPP_GRAPHGENERATOR_H
```

```
/*************************************************************************
                      GraphGenerator  -  description
                             -------------------
    begin                : 23/11/2015
    copyright            : (C) 2015 by Edern Haumont & Nicolas Six
*************************************************************************/

//-- Realisation of the class GraphGenerator (file GraphGenerator.cpp) ---

//---------------------------------------------------------------- INCLUDE

//-------------------------------------------------------- Personal include
#include "GraphGenerator.h"
#include "config.h"

// don't use: "using namespace std;" to keep clear that we use std and
// not any other library and by the same way keeping ready to use an other
// library than the std.

//----------------------------------------------------------------- PUBLIC

//--------------------------------------------------------- Public methods

// Constructor
GraphGenerator::GraphGenerator(const std::string &filePath)
{
    // Algorithm :
    // creating a new one graph file, deleting the older one if the file already exist
    graphFileStream.open(filePath,std::ios::out | std::ios::trunc);
    if(!graphFileStream)
    {
        std::cerr << "erreur lors de l'ouverture du fichier: " << filePath << std::endl;
    }
    else
    {
        graphFileStream << "digraph {" << std::endl;
    }
}
// Destructor
GraphGenerator::~GraphGenerator()
{
    if(graphFileStream)
    {
        graphFileStream << "}" << std::endl;
        graphFileStream.close();
    }
}

int GraphGenerator::addNodeToGraph(const std::string &nodeName)
{
    if(!graphFileStream)
    {
        std::cerr << "erreur sur le fichier en écriture" << std::endl;
        return FILE_ERROR;
    }
    else
    {
        graphFileStream << "        " << transformToNodeName(nodeName) << " [label=\"" << nodeName <<
"\"];" << std::endl;
    }
    return 0;
}
int GraphGenerator::addLinkToGraph(const std::string &nodeNameFrom, const std::string &nodeNameTo,
const std::string &linkLabel)
{
    addNodeToGraph(nodeNameFrom);
    addNodeToGraph(nodeNameTo);
    if(!graphFileStream)
    {
        std::cerr << "erreur sur le fichier en écriture" << std::endl;
        return FILE_ERROR;
    }
    else
    {
        graphFileStream << "        " << transformToNodeName(nodeNameTo) << " -> ";
        graphFileStream << transformToNodeName(nodeNameFrom) << " [label=" << linkLabel << "];" <<
std::endl;
    }
    return 0;
}

//---------------------------------------------------------- Private methods

std::string GraphGenerator::transformToNodeName(const std::string &nonUsableName) const
{
    std::string ret ="";
    bool add;
    for (unsigned i=0; i<nonUsableName.length(); i++)
    {
        add = true;
        for(unsigned n=0 ; n<INVALID_CHAR.length() ; n++)
        {
            if(nonUsableName.at(i)==INVALID_CHAR.at(n))
            {
                add = false;
                break;
            }
        }
        if(add)
        {
            ret.push_back(nonUsableName.at(i));
        }
        else
        {
            ret.push_back(nonUsableName.at(i)%26+'a');
        }
    }
    return ret;
}
```

```cpp
/*************************************************************************
                           LogOtherInfos  -  description
                             -------------------
    debut                : 23/11/2015
    copyright            : (C) 2015 by Edern Haumont & Nicolas Six
*************************************************************************/

#include <ctime>
#include <stdlib.h>
#include <string>
#include <iostream>


//------ Interface of the class LogOtherInfos (file LogOtherInfos.h) -----
#ifndef TP_CPP_LOGOTHERINFOS_H
#define TP_CPP_LOGOTHERINFOS_H

//------------------------------------------------------------- Constants


//------------------------------------------------------------------------
// This class contains all non classification informations
//------------------------------------------------------------------------

class LogOtherInfos
{
//----------------------------------------------------------------- PUBLIC
public:
//----------------------------------------------- Constructors - destructors
    LogOtherInfos();
    LogOtherInfos(const std::string &logIp,const tm &logTime,const unsigned int &logCode, const
unsigned &logSize,
                  const std::string &logBrowser, const std::string &logLogname, const std::string
&logPseudo,
                  const std::string &logRequest, const int &logGMT);
//-----------------------------------------------------------------Getter
    std::string & getRequest();
//---------------------------------------------------------------- PRIVATE
private:
//------------------------------------------------------- Private atributs
    std::string ip;
    tm time;
    unsigned int httpCode;
    unsigned sizeTransfered;
    std::string Browser;
    std::string logname;
    std::string pseudo;
    std::string request;
    int GMT;
};


#endif //TP_CPP_LOGOTHERINFOS_H
```

```
/*************************************************************************
                           LogOtherInfos  -  description
                             -------------------
    begin                : 23/11/2015
    copyright            : (C) 2015 by Edern Haumont & Nicolas Six
*************************************************************************/

//------ Realisation of the class LogOtherInfos (file LogOtherInfos) -----

//---------------------------------------------------------------- INCLUDE

//------------------------------------------------------- Personal include
#include "LogOtherInfos.h"

using namespace std;

//----------------------------------------------------------------- PUBLIC

//----------------------------------------------------------- Constructors
LogOtherInfos::LogOtherInfos()
{
}

LogOtherInfos::LogOtherInfos(const std::string &logIp,const tm &logTime,const unsigned int &logCode,
const unsigned &logSize,
                             const std::string &logBrowser, const std::string &logLogname, const
std::string &logPseudo,
                             const std::string &logRequest, const int &logGMT):
                    ip(logIp),
                    time(logTime),
                    httpCode(logCode),
                    sizeTransfered(logSize),
                    Browser(logBrowser),
                    logname(logLogname),
                    pseudo(logPseudo),
                    request(logRequest),
                    GMT(logGMT)
{
}

//getter
std::string & LogOtherInfos::getRequest() {
    return request;
}
```

```
/*************************************************************************
                             Configuration File
                             -------------------
    begin                : 23/11/2015
    copyright            : (C) 2015 by Edern Haumont & Nicolas Six
*************************************************************************/

#ifndef TP_CPP_CONFIG_H
#define TP_CPP_CONFIG_H

//--------------------------------------------------------------- Includes
#include <string>

//-------------------------------------------------------------- Constants
const std::string EXTENSION_FILE = "excludedExtension.txt";
const std::string INVALID_CHAR = "/\\:\"'^£$|[](){}#~?&%.=-+*,_ ";

#endif //TP_CPP_CONFIG_H
```