

```

1 #include <iostream>
2 #include <string>
3 #include <iomanip>
4
5 #include "ArgumentParser/ArgumentParser.h"
6 #include "Analyse/Analyse.h"
7
8 using namespace std;
9
10 int main(int argc, char *argv[])
11 {
12     Analyse analyse;
13     string command = " ";
14
15     for (int i = 1; i < argc; i++)
16         command.append(argv[i]).append(" ");
17
18     PARSE_RESULTS parseResult = ArgumentParser::Parse(
19         command, analyse);
20
21     if (parseResult == GOOD)
22     {
23         analyse.Run();
24     }
25     else if (parseResult == PARSING_ERROR)
26     {
27         cout
28
29             << endl;
30         cout << "Format de la commande non valide..." << endl;
31         cout
32
33             << endl;
34         cout << "Utilisation de ./analog [-e|-g|-t] <path>" << endl;
35         cout << "    <path>      : chemin vers un fichier << endl;
36         d'extension .txt ou .log." << endl;
37         cout << "    -e          : Exclut les documents de << endl;
38         type image, css ou javascript." << endl;
39         cout << "    -g <path.dot> : Génère le fichier <path << endl;
40         .dot> au format GraphViz du graphe analysé." << endl;
41         cout << "    -t [0-23]   : Prend en compte que les << endl;
42         hits dans le créneau horaire [t, t+1]." << endl;
43         cout
44
45             << endl;
46     }
47
48     return 0;

```

38 }

```
1 /*****  
2 * *****  
3 * Hit - description  
4 * -----  
5 * début : 11/12/2018  
6 * copyright : (C) 2018 par Valentin Gilles et  
7 * Balthazar Frolin  
8 * e-mail : ...@insa-lyon.fr  
9 * *****/  
10 //----- Interface de la classe <Hit> (fichier Hit.h )  
11 #define Hit_H  
12  
13 //-----  
14 // Interfaces utilisées  
15 #include "../Datetime/Datetime.h"  
16 #include "../Request/Request.h"  
17 #include <string>  
18 #include <iostream>  
19 using namespace std;  
20 //-----  
21 // Constantes  
22 // Change this value according to the server where you  
// record the logs.  
23 const static string SERVER_URL = "http://intranet-if.insa-  
lyon.fr";  
24  
25 //-----  
26 // Rôle de la classe <Hit>  
27 //  
28 // Permet de lire et de stocker une ligne de log.  
29 //-----  
30  
31 class Hit  
32 {  
33 //-----  
34 // PUBLIC  
35 public:  
36 //-----  
37 // Méthodes publiques  
38 const string & GetIp() const {
```

```

39         return ip;
40     }
41
42     const string & GetLogname() const {
43         return logname;
44     }
45
46     const string & GetAuthenticatedUser() const {
47         return authenticatedUser;
48     }
49
50     const Datetime & GetDatetime() const {
51         return datetime;
52     }
53
54     const Request & GetRequest() const {
55         return request;
56     }
57
58     unsigned int GetStatusCode() const {
59         return statusCode;
60     }
61
62     unsigned int GetDataQty() const {
63         return dataQty;
64     }
65
66     string GetReferer() const;
67
68     string GetRefererGetArgs() const {
69         return refererGetArgs;
70     }
71
72     const string & GetBrowserInfo() const {
73         return browserInfo;
74     }
75
76     bool IsRelatedToResourceFile() const {
77         return relatedToResourceFile;
78     }
79
80 //-----
81 // Surcharge d'opérateurs
82     friend istream & operator>>(istream & is, Hit & hit);
83     // Mode d'emploi :
84     // Fill hit instance with the first hit data found in
85     // is.
86     // Contrat :

```

```
87     // is content must be well formatted.
88     //
89
90 //-----
91     Constructeurs - destructeur
92     Hit() = default;
93     // Mode d'emploi :
94     // Empty Constructor
95     //
96     // Contrat :
97     //
98
99     virtual ~Hit() = default;
100    // Mode d'emploi :
101    // Empty Destructor
102    //
103    // Contrat :
104    //
105
106
107 //-----
108     ----- PRIVE
109 protected:
110 //-----
111     Méthodes protégées
112     void setDataQty(string &temp);
113     // Modd d'emploi:
114     // Check if temp contain DataQty else set to 0
115     //
116     // Contrat :
117     //
118
119     void setRefererInfos(string &temp);
120     // Modd d'emploi:
121     // Split URL Args from Referer.
122     //
123     // Contrat :
124     //
125
126     bool checkIfHitIsRelatedToAResourceFile(const string &
127     filePath);
128     // Mode d'emploi:
129     // Return true if file extension exist in
130     // RESOURCE_EXTENSION_LIST
131     //
132     // Contrat :
133     //
```

```
132 //-----  
133 //-----Attributs protégés  
134     string ip;  
135     string logname;  
136     string authenticatedUser;  
137  
138     Datetime datetime;  
139     Request request;  
140  
141     unsigned int statusCode;  
142     unsigned int dataQty;  
143  
144     string referer;  
145     string refererGetArgs;  
146  
147     string browserInfo;  
148     bool relatedToResourceFile;  
149 };  
150  
151 //-----Autres définitions  
152 //-----dépendantes de <Hit>  
153 #endif // Hit_H  
154  
155
```

```
1 /*****  
2 * *****  
3 * Hit - description  
4 * -----  
5 * début : 11/12/2018  
6 * copyright : (C) 2018 par Valentin Gilles et  
7 * Balthazar Frolin  
8 * e-mail : ...@insa-lyon.fr  
9 * *****/  
10 //----- Réalisation de la classe <Hit> (fichier Hit.  
11 //----- .cpp)  
12 //-----  
13 //----- INCLUDE  
14 //-----  
15 //----- Include système  
16 #include <iostream>  
17 //-----  
18 //----- Include personnel  
19 #include "Hit.h"  
20 //-----  
21 //----- Constantes  
22 const char BASIC_SEPARATOR = ' ' ;  
23 const char LONG_STRING_SEPARATOR = "" ;  
24 //-----  
25 const char * const RESOURCE_EXTENSION_LIST[] = {"jpg", "jpeg", "bmp", "tiff", "png", "gif", "css", "js", "ico"};  
26 //-----  
27 //----- PUBLIC  
28 //-----  
29 string Hit::GetReferer() const  
30 {  
31     if (referer.find(SERVER_URL) != -1)  
32     {  
33         return referer.substr(SERVER_URL.length(), string::npos);  
34     }  
35     else  
36     {  
37         return referer;  
38     }  
39 }
```

```

40
41 //-----
42 //----- Surcharge d'opérateurs
43 istream & operator>>(istream & is, Hit & hit)
44 {
45     string temp;
46
47     getline(is, hit.ip, BASIC_SEPARATOR);
48     getline(is, hit.logname, BASIC_SEPARATOR);
49     getline(is, hit.authenticatedUser, BASIC_SEPARATOR);
50
51     is >> hit.datetime;
52     is >> hit.request;
53
54     getline(is, temp, BASIC_SEPARATOR);
55     hit.statusCode = (unsigned int) stoi(temp, nullptr, 10)
56 ;
57
58     getline(is, temp, BASIC_SEPARATOR);
59     hit.setDataQty(temp);
60
61     is.seekg(1, ios_base::cur);
62     getline(is, temp, LONG_STRING_SEPARATOR);
63     hit.setRefererInfos(temp);
64
65     is.seekg(2, ios_base::cur);
66     getline(is, hit.browserInfo, LONG_STRING_SEPARATOR);
67
68     hit.relatedToResourceFile = hit.
69     checkIfHitIsRelatedToAResourceFile(hit.request.getUrl());
70
71     is.seekg(1, ios_base::cur);
72     if (is.get() != -1)
73         is.seekg(-1, ios_base::cur);
74
75     return is;
76 }
77
78 //-----
79 //----- PRIVE
80 //-----
81 //----- Méthodes protégées
82
83 void Hit::setDataQty(string &temp)
84 {
85     if (temp.c_str()[0] == '-')           //sometimes, field
86         data quantity has '-' char
87     {                                     //thus it's

```

```
83 necessary to test it manually to avoid stoi errors
84         dataQty = 0;
85     }
86 else
87 {
88     dataQty = (unsigned int) stoi(temp, nullptr, 10);
89 }
90 }
91
92 void Hit::setRefererInfos(string &temp)
93 {
94     unsigned long argsGetpos = temp.find('\'');
95     if (argsGetpos == string::npos)
96         argsGetpos = temp.find_first_of('?');
97
98     if (argsGetpos != string::npos)
99     {
100         referer = temp.substr(0, argsGetpos);
101         refererGetArgs = temp.substr(argsGetpos + 1);
102     }
103 else
104 {
105     referer = temp;
106 }
107
108 if (referer.back() == '/')
109     referer.erase(referer.end() - 1, referer.end());
110 }
111
112 bool Hit::checkIfHitIsRelatedToAResourceFile(const string
&filePath)
113 {
114     unsigned long dotPosition = filePath.find_last_of('.');
115 ;
116     if (dotPosition != -1)
117     {
118         string extension = filePath.substr(dotPosition + 1);
119         for (auto &i : RESOURCE_EXTENSION_LIST)
120         {
121             if (extension == i)
122                 return true;
123         }
124     }
125
126     return false;
127 }
```

```

1 ****
2          Analyse - description
3
4      début           : 11/12/2018
5      copyright       : (C) 2018 par Valentin Gilles et
6      Balthazar Frolin
7      e-mail          : ...@insa-lyon.fr
8 ****
9 //----- Interface de la classe <Analyse> (fichier
10 #if ! defined ( Analyse_H )
11 #define Analyse_H
12
13 //-----
14 //----- Interfaces utilisées
15 #include "../GraphVizWriter/GraphVizWriter.h"
16 #include "../LogReader/LogReader.h"
17
18 #include <utility>
19
20 using namespace std;
21
22 //-----
23 //----- Constantes
24 //-----
25 //----- Types
26
27 //----- Rôle de la classe <Analyse>
28 //----- Classe principale de l'application, permet de créer les
29 //----- structures de données
30 //----- utilisé pour créer un top 10 et pour créer le graph.
31 //----- Class
32
33 //----- class Analyse
34 {
35 //-----
```

```

39 ----- PUBLIC
40
41 public:
42 //-----
43     Méthodes publiques
44
45     void Run();
46     // Mode d'emploi :
47     // Run Analysis on log, generate graph if asked and
48     // display top 10.
49     //
50     // Contrat :
51     //
52
53     void SetGraph(const string & graphPath)
54     {
55         this->generateGraph = true;
56         this->graphPath = graphPath;
57     }
58
59     void SetHour(int hour)
60     {
61         this->hour = hour;
62     }
63
64     void SetExcludeResourcesFile(bool excludeResourcesFile)
65     {
66         this->excludeResourcesFile = excludeResourcesFile;
67     }
68
69     void SetLogReader(LogReader * logReader)
70     {
71         this->logReader = logReader;
72     }
73
74 //-----
75     Constructeurs – destructeur
76
77     Analyse ():hour(-1), excludeResourcesFile(false),
78     generateGraph(false) {};
79     // Mode d'emploi :
80     // default constructor
81     //
82     // Contrat :
83     //
84     virtual ~Analyse () = default;
85     // Mode d'emploi :
86     // default destructor
87     //

```

```
85      // Contrat :
86      //
87
88 //-----
89      ----- PRIVE
90
91 protected:
92 //-----
93      Méthodes protégées
94
95      bool analyseNextHit();
96      // Mode d'emploi :
97      // Get next Hit and add it to NodeCounter if it match
98      // analysis parameters.
99      //
100
101     const string * updateNodeCounterMapWithUrl(const
102         string &url);
103     // Mode d'emploi :
104     // Add an entry or increment NodeCounter
105     //
106     // Contrat :
107     //
108     const string * getRefererStringInNodeCounterMap(const
109         string &referer);
110     // Mode d'emploi :
111     // Get Referer String pointer from counter map or
112     // insert a new one if it don't exist.
113     //
114     void updateGraphMapper(const string * referer, const
115         string * url);
116     // Mode d'emploi :
117     // Add an entry or increment GraphMapper
118     //
119     // Contrat :
120     //
121     void generateOrderedNodeCounterMap();
122     // Mode d'emploi :
123     // Generate Ordered Node Counter by inverting
124     // nodeCounter Map.
125     //
126     // Contrat :
```

```
127
128     void displayResult();
129     // Mode d'emploi :
130     // Display a top ten from orderedNodeCounterMap.
131     //
132     // Contrat :
133     //
134
135 //-----  
    Attributs protégés
136
137     string graphPath;
138     int hour;
139     bool excludeResourcesFile;
140     bool generateGraph;
141
142     LogReader * logReader;
143
144     GraphMapper graphMapper;
145     NodeCounter nodeCounter;
146     OrderedNodeCounter orderedNodeCounter;
147 };
148
149 //----- Autres définitions  
dépendantes de <Analyse>
150
151 #endif // Analyse_H
152
153
```

```
1 ****
2          Analyse - description
3
4      début           : 11/12/2018
5      copyright       : (C) 2018 par Valentin Gilles et
6      Balthazar Frolin
7      e-mail          : ...@insa-lyon.fr
8 ****
9 //----- Réalisation de la classe <Analyse> (fichier
10 //----- Analyse.cpp) -----
11 //-----
12 //----- INCLUDE
13 //-----
14 //----- Include système
15 #include <iostream>
16 #include <iomanip>
17 //-----
18 //----- Include personnel
19 #include "Analyse.h"
20 using namespace std;
21 //-----
22 //----- Constantes
23 //-----
24 //----- PUBLIC
25 //-----
26 //----- Méthodes publiques
27 void Analyse::Run()
28 {
29     while(analyseNextHit());
30
31     generateOrderedNodeCounterMap();
32
33     if (generateGraph)
34     {
35         if (GraphVizWriter::Write(nodeCounter, graphMapper,
36             graphPath))
37         {
38             cout << "Le Fichier WizGraph a bien été généré
. " << endl;
39         }
40     }
41 }
```

```

39         else
40         {
41             cout << "Le Fichier WizGraph n'a pas pu être
42         générée." << endl;
43     }
44     displayResult();
45 }
46
47 //-----
48 //----- Surcharge d'opérateurs
49 //-----
50 //----- Constructeurs – destructeur
51 //-----
52 //----- PRIVE
53 //-----
54 //----- Méthodes protégées
55 bool Analyse::analyseNextHit()
56 {
57     Hit * hitPtr;
58     hitPtr = logReader->ReadNext();
59
60     if (hitPtr)
61     {
62         if ((hour == -1 || (hitPtr->GetDatetime().GetHour()
63 == hour))
64             && (!excludeResourcesFile || !hitPtr->
65             IsRelatedToResourceFile()))
66         {
67             const string * url =
68             updateNodeCounterMapWithUrl(hitPtr->GetRequest().getUrl());
69
70             if (generateGraph)
71             {
72                 const string * referer =
73                 getRefererStringInNodeCounterMap(hitPtr->GetReferer());
74                 updateGraphMapper(referer, url);
75
76                 delete hitPtr;
77             }
78         }
79         return true;
80     }
81     return false;
82 }
83
84 const string * Analyse::getRefererStringInNodeCounterMap(

```

```

79 const string &referer)
80 {
81     auto nodeReferer = nodeCounter.find(referer);
82
83     if (nodeReferer == nodeCounter.end()) {
84         nodeReferer = nodeCounter.insert({referer, 0}).
85         first;
86     }
87     const string * ptrToReferer = &(nodeReferer->first);
88
89     return &(nodeReferer->first);
90 }
91
92 const string * Analyse::updateNodeCounterMapWithUrl(const
93     string &url)
94 {
95     auto nodeCounterResultUrl = nodeCounter.find(url);
96
97     if (nodeCounterResultUrl != nodeCounter.end())
98     {
99         (nodeCounterResultUrl->second)++;
100    }
101
102    else
103    {
104        nodeCounter.insert({url, 1});
105        nodeCounterResultUrl = nodeCounter.find(url);
106    }
107
108    return &(nodeCounterResultUrl->first);
109 }
110
111 void Analyse::updateGraphMapper(const string * referer,
112     const string * url)
113 {
114     auto graphMapperResult = graphMapper.find(pair<const
115         string *, const string *>(referer, url));
116     if (graphMapperResult != graphMapper.end())
117     {
118         (graphMapperResult->second)++;
119     }
120     else
121     {
122         graphMapper.insert({pair<const string *, const
123             string *>(referer, url), 1});
124     }
125 }
126
127 void Analyse::generateOrderedNodeCounterMap()
128 {
129     for (auto &it : nodeCounter)
130     {

```

```
124         orderedNodeCounter.insert({it.second, &(it.first)})
125     }
126 }
127
128 void Analyse::displayResult()
129 {
130     unsigned int i = 1;
131
132     for (auto it = orderedNodeCounter.rbegin(); it != orderedNodeCounter.rend() && i <= 10; it++)
133     {
134         cout << setw(2) << i << " - Cible : " << *(it).second << " : " << (*it).first << " visite(s)" << endl;
135         i++;
136     }
137 }
138
```

```

1 /***** Request - description *****/
2
3
4     début          : 11/12/2018
5     copyright      : (C) 2018 par Valentin Gilles et
6     Balthazar Frolin
7     e-mail         : ...@insa-lyon.fr
8
9 //----- Interface de la classe <Request> (fichier Request
10 .h)
11 #if ! defined ( Request_H )
12 #define Request_H
13 //-----
14 //----- Interfaces utilisées
15 #include <string>
16 #include <iostream>
17
18 using namespace std;
19 //-----
20 //-----
21 // Rôle de la classe <Request>
22 //
23 // Permet de lire et de stocker un objet Request.
24 //
25 //-----
26
27 class Request
28 {
29 //-----
30 //----- PUBLIC
31 public:
32 //-----
33 //----- Méthodes publiques
34     const string & getType() const {
35         return type;
36     }
37
38     const string & getUrl() const {
39         return url;
40     }

```

```

41     const string & getUrlGetArgs() const {
42         return urlGetArgs;
43     }
44
45
46     const string & getProtocol() const {
47         return protocol;
48     }
49
50 //-----
51 //----- Surcharge d'opérateurs
52     friend istream & operator>>(istream & is, Request &
53     request);
54
55 //----- Constructeurs - destructeur
56     Request() = default;
57     // Mode d'emploi :
58     // Empty Constructor
59     //
60     // Contrat :
61     //
62     virtual ~Request() = default;
63     // Mode d'emploi :
64     // Empty Destructor
65     //
66     // Contrat :
67     //
68
69 //----- PRIVE
70
71 protected:
72
73     void setUrlInfos(string &temp);
74
75
76 //----- Attributs protégés
77
78     string type;
79     string url;
80     string urlGetArgs;
81     string protocol;
82 };
83
84 //----- Autres définitions

```

```
84 dépendantes de <Request>
85
86 #endif // Request_H
87
88
```

```
1 /*****  
2 * Request - description  
3 *-----  
4   début          : 11/12/2018  
5   copyright      : (C) 2018 par Valentin Gilles et  
6   Balthazar Frolin  
7   e-mail         : ...@insa-lyon.fr  
8 *****/  
9 //----- Réalisation de la classe <Request> (fichier  
10 Request.cpp) -----  
11 //-----  
12 ----- INCLUDE  
13 //-----  
14 ----- Include système  
15 #include <iostream>  
16 //-----  
17 ----- Include personnel  
18 #include "Request.h"  
19 //-----  
20 using namespace std;  
21 //-----  
22 ----- Constantes  
23 const char BASIC_SEPARATOR = ' ' ;  
24 const char LONG_STRING_SEPARATOR = "\000" ;  
25 //-----  
26 ----- PUBLIC  
27 //-----  
28 ----- Surcharge d'opérateurs  
29 //-----  
30 istream & operator>>(istream & is, Request & request)  
31 {  
32     string temp, urlTemp;  
33     is.seekg(2, ios_base::cur);  
34     getline(is, request.type, BASIC_SEPARATOR);  
35     getline(is, temp, LONG_STRING_SEPARATOR);  
36     unsigned long lastSpacePos = temp.rfind(' ');  
37 }
```

```
41     request.protocol = temp.substr(lastSpacePos + 1);
42
43     temp = temp.substr(0, lastSpacePos);
44     request.setUrlInfos(temp);
45
46     is.seekg(1, ios_base::cur);
47
48     return is;
49 }
50
51 //-----
52      ----- PRIVE
53 //-----
54
55 void Request::setUrlInfos(string &temp) // TODO: Create new
56 // Object for URL to remove code redundancy with Hit Class
57 {
58     unsigned long argsGetpos = temp.find('\'');
59     if (argsGetpos == string::npos)
60         argsGetpos = temp.find_first_of('?');
61
62     if (argsGetpos != string::npos)
63     {
64         url = temp.substr(0, argsGetpos);
65         urlGetArgs = temp.substr(argsGetpos + 1);
66     }
67     else
68     {
69         url = temp;
70     }
71
72     if (url.back() == '/' && url.size() > 1)
73         url.erase(url.end() - 1, url.end());
74 }
```

```
1 /*****  
2 * *****  
3 * Datetime - description  
4 * -----  
5 * début : 11/12/2018  
6 * copyright : (C) 2018 par Valentin Gilles et  
7 * Balthazar Frolin  
8 * e-mail : ...@insa-lyon.fr  
9 * *****/  
10 //----- Interface de la classe <Datetime> (fichier Datetime  
11 .h)  
12 #if ! defined ( Datetime_H )  
13 #define Datetime_H  
14 //-----  
15 // Interfaces utilisées  
16 #include <string>  
17 #include <iostream>  
18 //-----  
19 // Constantes  
20 const unsigned short int MONTH_LENGTH = 3;  
21 //-----  
22 //-----  
23 // Rôle de la classe <Datetime>  
24 //  
25 // Permet de lire et de stocker un objet datetime.  
26 //-----  
27  
28 class Datetime  
29 {  
30 //-----  
31 //----- PUBLIC  
32 public:  
33  
34 //-----  
35 //----- Methode publics  
36     unsigned short GetDay() const {  
37         return day;  
38     }  
39  
40     const char * GetMonth() const {
```

```
41         return month;
42     }
43
44     unsigned short GetYear() const {
45         return year;
46     }
47
48     unsigned short GetHour() const {
49         return hour;
50     }
51
52     unsigned short GetMinutes() const {
53         return minutes;
54     }
55
56     unsigned short GetSecondes() const {
57         return seconds;
58     }
59
60     const string& GetUtc() const {
61         return utc;
62     }
63
64 //-----
65 //----- Surcharge d'opérateurs
66 friend istream & operator>>(istream & is, Datetime & datetime);
67
68 //-----
69 //----- Constructeurs – destructeur
70
71     Datetime() = default;
72     // Mode d'emploi :
73     // Empty Constructor
74     //
75     // Contrat :
76     //
77     virtual ~Datetime() = default;
78     // Mode d'emploi :
79     // Empty Destructor
80     //
81     // Contrat :
82     //
83
84 //-----
85 //----- PRIVE
86
87 protected:
```

```
87 //---  
88     Méthodes protégées  
89 //---  
90     Attributs protégés  
91     unsigned short int day;  
92     char month[MONTH_LENGTH + 1];  
93     unsigned short int year;  
94     unsigned short int hour;  
95     unsigned short int minutes;  
96     unsigned short int seconds;  
97  
98     string utc;  
99 };  
100  
101 //----- Autres définitions  
102     dépendantes de <Datetime>  
103 #endif // Datetime_H
```

```

1 /***** ****
2                               Datetime - description
3
4   début           : 11/12/2018
5   copyright       : (C) 2018 par Valentin Gilles et
6   Balthazar Frolin
7   e-mail          : ...@insa-lyon.fr
8 *****/
9 //---- Réalisation de la classe <Datetime> (fichier
10 //----- Datetime.cpp) -----
11 //-----
12 //----- INCLUDE
13 //-----
14 //----- Include système
15 #include <iostream>
16 //-----
17 //----- Include personnel
18 #include "Datetime.h"
19 using namespace std;
20 //-----
21 //----- Constantes
22 const char DATE_SEPARATOR = '/';
23 const char TIME_SEPARATOR = ':';
24 const char BASIC_SEPARATOR = ' ';
25 const char END_INDICATOR = ']';
26
27 //-----
28 //----- PUBLIC
29 //-----
30 //----- Méthodes publiques
31 //-----
32 //----- Surcharge d'opérateurs
33 istream & operator>>(istream & is, Datetime & datetime)
34 {
35     string temp;
36
37     is.seekg(1, ios_base::cur);           // Escape
38     First Start Indicator

```

```
39     getline(is, temp, DATE_SEPARATOR);
40     datetime.day = (unsigned short) stoul(temp, nullptr, 10)
41 ;
42     getline(is, temp, DATE_SEPARATOR);
43     sprintf(datetime.month, "%4s", temp.c_str());
44
45     getline(is, temp, TIME_SEPARATOR);
46     datetime.year = (unsigned short) stoul(temp, nullptr, 10)
47 );
48     getline(is, temp, TIME_SEPARATOR);
49     datetime.hour = (unsigned short) stoul(temp, nullptr, 10)
50 );
51     getline(is, temp, TIME_SEPARATOR);
52     datetime.minutes = (unsigned short) stoul(temp, nullptr
53 , 10);
54     getline(is, temp, BASIC_SEPARATOR);
55     datetime.seconds = (unsigned short) stoul(temp, nullptr
56 , 10);
57     getline(is, datetime.utc, END_INDICATOR);
58
59     return is;
60 }
61
62 //-----
63 //----- PRIVE
64 //-----
65 //----- Méthodes protégées
66
```

```
1 /*****  
2 * *****/  
3  
4     début           : 11/12/2018  
5     copyright       : (C) 2018 par Valentin Gilles et  
6     Balthazar Frolin  
7     e-mail          : ...@insa-lyon.fr  
8  
9 //---- Interface de la classe <LogReader> (fichier  
10 //---- LogReader.h) -----  
11 #if ! defined ( LogReader_H )  
12 #define LogReader_H  
13 //-----  
14 //----- Interfaces utilisées  
15 #include <fstream>  
16 #include "../Hit/Hit.h"  
17  
18 using namespace std;  
19 //-----  
20 //----- Constantes  
21 //-----  
22 //----- Types  
23 //-----  
24 // Rôle de la classe <LogReader>  
25 //  
26 // La classe LogReader permet de charger et de lire le  
// contenu d'un fichier  
27 // de log en créant des objets Hit pour chaque entrée.  
28 //  
29 //-----  
30  
31 class LogReader  
32 {  
33 //-----  
34 //----- PUBLIC  
35 public:  
36 //-----  
37 //----- Méthodes publiques  
38     Hit * ReadNext();
```

```
39     // Mode d'emploi :  
40     // Read Next Log from stream.  
41     // return nullptr if stream is ended.  
42     // else return an instance of the Hit created.  
43     //  
44     // Contrat:  
45     //  
46  
47     bool TrackNewFile(const string & path);  
48     // Mode d'emploi :  
49     // Close the old stream and try to open a new one at  
path  
50     // Return true if file opened successfully else return  
false.  
51     //  
52     // Contrat:  
53     //  
54  
55     void CloseFileStream();  
56     // Mode d'emploi :  
57     // Close stream if opened.  
58     //  
59     // Contrat:  
60     //  
61  
62 //-----  
63     Surchage d'opérateurs  
64 //-----  
65     Constructeurs – destructeur  
66     explicit LogReader(const string & path);  
67     // Mode d'emploi :  
68     // Instantiate LogReader and open stream at path.  
69     //  
70     // Contrat :  
71     //  
72  
73     virtual ~LogReader();  
74     // Mode d'emploi :  
75     // close istream.  
76     //  
77     // Contrat :  
78     //  
79  
80 //-----  
----- PRIVE  
81  
82 protected:  
83
```

```
84 //-----  
85     Attributs protégés  
86     ifstream stream;  
87  
88 };  
89  
90 //----- Autres définitions  
91     dépendantes de <LogReader>  
92 #endif // LogReader_H  
93  
94
```

```

1 /*****
2 * LogReader - description
3 *
4 * début : 11/12/2018
5 * copyright : (C) 2018 par Valentin Gilles et
6 * Balthazar Frolin
7 * e-mail : ...@insa-lyon.fr
8 *****/
9 //---- Réalisation de la classe <LogReader> (fichier
10 // LogReader.cpp) -----
11 //-----
12 //----- INCLUDE
13 //-----
14 //----- Include système
15 #include <iostream>
16 #include <exception>
17 //-----
18 //----- Include personnel
19 #include "LogReader.h"
20 using namespace std;
21 //-----
22 //----- PUBLIC
23 //-----
24 //----- Méthodes publiques
25
26 Hit * LogReader::ReadNext()
27 {
28     if (stream.is_open() && !stream.eof())
29     {
30         Hit * hit = new Hit;
31         stream >> *hit;
32
33         return hit;
34     }
35     return nullptr;
36 }
37
38 bool LogReader::TrackNewFile(const string &path)
39 {
40     CloseFileStream();
41

```

```
42     stream.open(path, ios_base::in);
43
44     return(stream.good());
45 }
46
47 void LogReader::CloseFileStream()
48 {
49     if (stream.is_open())
50         stream.close();
51 }
52
53 //-----
Constructeurs – destructeur
54
55 LogReader::LogReader(const string & path)
56 {
57 #ifdef MAP
58     cout << "Appel au destructeur de <LogReader>" << endl;
59 #endif
60
61     if (!TrackNewFile(path))
62         throw invalid_argument("Error Opening File : " +
path);
63 } //---- Fin de LogReader
64
65 LogReader::~LogReader()
66 {
67 #ifdef MAP
68     cout << "Appel au constructeur de <LogReader>" << endl;
69 #endif
70
71     CloseFileStream();
72 } //---- Fin de ~LogReader
73
74
75 //-----
Méthodes protégées
76
77
```

```
1 /*****  
2 * **** ArgumentParser - description  
3 * -----  
4 * début : 11/12/2018  
5 * copyright : (C) 2018 par Valentin Gilles et  
6 * Balthazar Frolin  
7 * e-mail : ...@insa-lyon.fr  
8 * *****/  
9 // Interface de la classe <ArgumentParser> (fichier  
10 // ArgumentParser.h)  
11 #if ! defined ( ArgumentParser_H )  
12 #define ArgumentParser_H  
13 //-----  
14 //----- Interfaces utilisées  
15 #include <regex>  
16  
17 #include "../LogReader/LogReader.h"  
18 #include "../Analyse/Analyse.h"  
19  
20 //-----  
21 //----- Constantes  
22 //-----  
23 //----- Types  
24 enum PARSE_RESULTS {  
25     GOOD,  
26     FILE_NO_OVERRIDE,  
27     FILE_NOT_FOUND,  
28     PARSING_ERROR,  
29     INVALID_VALUE,  
30 };  
31  
32 //-----  
33 // Rôle de la classe <ArgumentParser>  
34 //  
35 // Parse la commande en entrée et configure l'instance d'  
// Analyse associée  
36 //  
37 //-----  
38  
39 class ArgumentParser  
40 {
```

```
41 //-----  
42     ----- PUBLIC  
43 public:  
44 //-----  
45     Méthodes publiques  
46     static PARSE_RESULTS Parse(string &args, Analyse &  
        analyse);  
47     // Mode d'emploi :  
48     // Check if args are well formatted and setup analyse  
        according to them.  
49     //  
50     // args      : string      command args.  
51     // analyse   : Analyse    analyse object to be configured  
52     ..  
53     // return    : Boolean    true if args are well formatted  
        , else false.  
54     //  
55 //-----  
56     Constructeurs – destructeur  
56  
57     ArgumentParser() = delete;  
58  
59 //-----  
60     ----- PRIVE  
61 protected:  
62 //-----  
63     Méthodes protégées  
64  
65     static bool testTimeArgs(string &args, Analyse &analyse  
        );  
66     // Mode d'emploi :  
67     // Check temps args formatting and configure analyse  
        according to it.  
68     //  
69     // return : true if args is well formatted, false sinon  
70     .  
71     //  
72     static bool testGraphArgs(string &args, Analyse &  
        analyse);  
73     // Mode d'emploi :  
74     // Check graph generation args formatting and configure  
        analyse according to it.  
75     //  
76     // return : true if args is well formatted, false sinon
```

```
76 .
77     //
78
79     static bool testLogArgs(string &args, Analyse &analyse
80 );
81     // Mode d'emploi :
82     // Check log path args formatting and configure
83     analyse according to it.
84     //
85
86     static bool askForFileOverride();
87     // Mode d'emploi :
88     // Ask user to confirm if he want to override file (
89     // with O/N)
90     //
91     // return : true or false according to user answer.
92 //-----
93     Attributs protégés
94
95     static const regex commandRegex;
96     static const regex graphPathArgRegex;
97     static const regex timeArgRegex;
98     static const regex logPathArgRegex;
99
100 //----- Autres définitions dépendantes de
101 // <ArgumentParser>
102 #endif // ArgumentParser_H
103
104
```

```

1 /*****
2          ****
3          ****
4      début           : 11/12/2018
5      copyright       : (C) 2018 par Valentin Gilles et
6      Balthazar Frolin
7      e-mail          : ...@insa-lyon.fr
8 ****
9 // Réalisation de la classe <ArgumentParser> (fichier
10 // ArgumentParser.cpp)
11 //-----
12 ----- INCLUDE
13 //-----
14 ----- Include système
15 #include <iostream>
16 #include <string>
17 //
18 //-----
19 ----- Include personnel
20 #include "ArgumentParser.h"
21 //
22 using namespace std;
23 //-----
24 ----- Constantes
25 const regex ArgumentParser::commandRegex      = regex(R"(
26 ^(( -g \S+\.\dot)|( -e)|( -t [0-9]+)){0,3} \S+\.(txt|log) $")
27 ")");
28 const regex ArgumentParser::graphPathArgRegex = regex("\\"S+
29 \\.\dot");
30 const regex ArgumentParser::timeArgRegex      = regex("-t [0-9]+");
31 const regex ArgumentParser::logPathArgRegex   = regex("\\"S+
32 \\.(txt|log)");
33 //
34 ----- PUBLIC
35 //
36 ----- Méthodes publiques
37 //
38 PARSE_RESULTS ArgumentParser::Parse(string &args, Analyse &
39 analyse)

```

```

35 {
36     if (!regex_match(args, commandRegex))
37         return PARSING_ERROR;
38
39     if (!testTimeArgs(args, analyse))
40         return INVALID_VALUE;
41
42     if (!testGraphArgs(args, analyse))
43         return FILE_NO_OVERRIDE;
44
45     if (args.find("-e") != string::npos)
46         analyse.SetExcludeResourcesFile(true);
47
48     if (!testLogArgs(args, analyse))
49         return FILE_NOT_FOUND;
50
51     return GOOD;
52 }
53
54 //-----
55 //----- PRIVE
56 //-----
57 //----- Méthodes protégées
58 bool ArgumentParser::testTimeArgs(string &args, Analyse &
59 {
60     smatch match;
61
62     if (regex_search(args, match, timeArgRegex))
63     {
64         string arg = match[0];
65         size_t pos = arg.find(' ');
66         int hour;
67
68         try
69         {
70             hour = stoi(arg.substr(pos));
71         }
72         catch(exception & e)
73         {
74             return false;
75         }
76
77         if (hour < 0 || hour > 23)
78         {
79             cout << "L'heure doit être un entier compris
entre 0 et 23." << endl;
80             return false;

```

```
81         }
82
83     analyse.SetHour(hour);
84 }
85
86     return true;
87 }
88
89 bool ArgumentParser::testGraphArgs(string &args, Analyse &
90 {
91     smatch match;
92
93     if (regex_search(args, match, graphPathArgRegex))
94     {
95         string path = match[0];
96
97         ifstream fileExistStream(path);
98         if (fileExistStream.good())
99         {
100             fileExistStream.close();
101
102             if (askForFileOverride())
103             {
104                 analyse.SetGraph(path);
105                 return true;
106             }
107             else
108             {
109                 cout << "Analyse de log annulé, veuillez
110 choisir un autre fichier pour le graph ou désactiver cette
111 option.";
112             }
113
114             ofstream fileCanBeCreatedStream(path);
115             if (fileCanBeCreatedStream.good())
116             {
117                 analyse.SetGraph(path);
118                 return true;
119             }
120
121             cout << "Le chemin spécifié pour la création du
122 fichier .dot n'est pas valide..." << endl;
123             return false;
124
125     return true;
126 }
```

```
127 bool ArgumentParser::askForFileOverride()
128 {
129     for ( ; ; )
130     {
131         char r;
132         cout << "Le fichier existe déjà, voulez vous l' écraser, (o/n) : ";
133         cin >> r;
134
135         if (r == 'o' || r == 'O')
136             return true;
137
138         if (r == 'n' || r == 'N')
139             return false;
140     }
141 }
142
143 bool ArgumentParser::testLogArgs(string &args, Analyse & analyse)
144 {
145     smatch match;
146
147     if (regex_search(args, match, logPathArgRegex))
148     {
149         LogReader * logReader;
150
151         try
152         {
153             logReader = new LogReader(match[0]);
154         }
155         catch (exception & e)
156         {
157             cerr << "Impossible d'ouvrir le fichier demandé." << endl;
158             return false;
159         }
160
161         analyse.SetLogReader(logReader);
162         return true;
163     }
164
165     return true;
166 }
```

```

1 /***** ****
2                               GraphVizWriter - description
3
4   début           : 11/12/2018
5   copyright       : (C) 2018 par Valentin Gilles et
6   Balthazar Frolin
7   e-mail          : ...@insa-lyon.fr
8 *****/
9 //-- Interface de la classe <GraphVizWriter> (fichier
10 //-- GraphVizWriter.h) --
11 #if ! defined ( GraphVizWriter_H )
12 #define GraphVizWriter_H
13 //-----
14 //----- Interfaces utilisées
15 #include <string>
16 #include <map>
17 #include <unordered_map>
18
19 using namespace std;
20
21 //-----
22 //----- Types
23 struct PairHash {
24 public:
25     template <typename T, typename U>
26     size_t operator()(const pair<T, U> &x) const
27     {
28         return hash<T>()(x.first) ^ hash<U>()(x.second);
29     }
30 };
31
32 typedef unordered_map<pair<const string *, const string *>, unsigned int, PairHash> GraphMapper;
33
34 //-----
35 // Rôle de la classe <GraphVizWriter>
36 //
37 // Classe static permettant de créer un fichier GraphViz
38 // depuis un objet de type GraphMapper
39 //
40 //-----
41

```

```
42 class GraphVizWriter
43 {
44 //-----
45 //----- PUBLIC
46 public:
47 //-----
48 //----- Méthodes publiques
49     static bool Write(unordered_map<string, unsigned int> &
50                     nodeCounterMap, GraphMapper &graphMapper, string &path);
51     // Mode d'emploi:
52     // Create file at path from nodeCounterMap &
53     graphMapper
54     //
55     // Contrat:
56     // path must be a valid file path.
57     //
58 //-----
59     -- Constructeur
60     GraphVizWriter() = delete;
61 };
62
63 //----- Autres définitions dépendantes de <
64 //----- GraphVizWriter>
65 #endif // GraphVizWriter_H
66
67
```

```

1 ****
2 ****
3 ****
4     début          : 11/12/2018
5     copyright      : (C) 2018 par Valentin Gilles et
6     Balthazar Frolin
7     e-mail         : ...@insa-lyon.fr
8 ****
9 // Réalisation de la classe <GraphVizWriter> (fichier
10 // GraphVizWriter.cpp)
11 //-----
12 ----- INCLUDE
13 //-----
14 ----- Include système
15 #include <iostream>
16 #include <fstream>
17 #include <sstream>
18
19 //-----
20 ----- Include personnel
21 #include "GraphVizWriter.h"
22
23 using namespace std;
24
25 //-----
26 ----- PUBLIC
27 //-----
28 ----- Méthodes publiques
29 bool GraphVizWriter::Write(unordered_map<string, unsigned
30 int> &nodeCounterMap, GraphMapper &graphMapper, string &
31 path)
32 {
33     ofstream stream(path);
34     if (!stream.good())
35         return false;
36
37     stream << "digraph {" << endl;
38
39     for (auto &it : nodeCounterMap)
40     {
41         stream << "ID" << &it.first << " [label=\""
42         << it.

```

```
39 first << "\"" ; " << endl;
40     }
41
42     for (auto &it : graphMapper)
43     {
44         stream << "ID" << it.first.first << " -> " << "ID"
45         << it.first.second << " [label=\"" << it.second << "\"];"
46         << endl;
47     }
48
49     stream << "}" << endl;
50
51     return true;
52 }
```