

```

1  #include "CsvParser.hpp"
2  #include "Arp/System/Commons/Logging.h"
3  #include "Arp/System/Core/ByteConverter.hpp"
4
5  namespace Parser
6  {
7
8  void CsvParser::Execute()
9  {
10     // File to be read
11     std::string fileToRead = this->fileLocation + this->fileName;
12     if (true == this->readFile) {
13         ifstream file;
14         file.open(fileToRead, ios::in);
15
16         if (file.is_open()) {
17             std::string line;
18
19             // Ignore the first line (header) of the file
20             if (-1 != std::getline(file, line)) {
21                 // Structure with single line data fields
22                 strSingleData info;
23                 // Array with all data (all rows of the file)
24                 arrPlant plant;
25
26                 // counters for ports
27                 int16 counterPort1 = 0, counterPort2 = 0, counterPort3 = 0,
28                 counterPort4 = 0;
29                 int lineNumber = 0;
30
31                 while (std::getline(file, line)) {
32                     this->log.Info("Processing line number {0}", lineNumber);
33                     std::stringstream row(line);
34                     std::string data;
35
36                     for (int columna = 0; std::getline(row, data, ',');
37                         ++columna) {
38                         switch (columna) {
39                             case HeaderValue::E_MODE:
40                                 if (data == "RTU") {
41                                     info.iModbusType = 0;
42                                 } else if (data == "TCP") {
43                                     info.iModbusType = 1;
44                                 } else {
45                                     info.iModbusType = -1;
46                                 }
47                                 break;
48                             case HeaderValue::E_ADDRESS:
49                                 info.uiSlaveNo = std::stoi(data);
50                                 break;
51                             case HeaderValue::E_FUNCTION:
52                                 info.iFc = std::stoi(data);
53                                 break;
54                             case HeaderValue::E_BASE_REGISTER:
55                                 info.uiAddress = std::stoi(data);
56                                 break;
57                             case HeaderValue::E_NUM_OF_REGISTERS:
58                                 info.iQtyRegs = std::stoi(data);
59                                 break;
60                             case HeaderValue::E_OPCUA_INDEX:
61                                 info.iOpcUaIndex = std::stoi(data);
62                                 break;
63                             default:
64                                 break;
65                         }
66                     }
67
68                     // Manage data by port number
69                     insertIntoPlantArray(info, plant, counterPort1,
70                     counterPort2, counterPort3, counterPort4);
71                     lineNumber++;
72                 }
73             }
74         }
75     }
76 }

```

```

73     //                printPlant(plant);
74
75         file.close();
76     } else {
77         Log::Error("CsvParser: File {0} is empty.", fileToRead);
78     }
79 } else {
80     Log::Error("CsvParser: Error opening file: {0}.", fileToRead);
81 }
82 } else {
83     Log::Info("CsvParser: The flag to read the file is deactivated");
84 }
85 }
86
87 void CsvParser::insertIntoPlantArray(const strSingleData &info,
88                                     arrPlant &planta,
89                                     const int16 &cport1,
90                                     const int16 &cport2,
91                                     const int16 &cport3,
92                                     const int16 &cport4)
93 {
94     switch (info.iModbusType)
95     {
96     case 0:
97         planta[info.iModbusType][cport1] = info;
98         cport1++;
99         break;
100    case 1:
101        planta[info.iModbusType][cport2] = info;
102        cport2++;
103        break;
104    case 2:
105        planta[info.iModbusType][cport3] = info;
106        cport3++;
107        break;
108    case 3:
109        planta[info.iModbusType][cport4] = info;
110        cport4++;
111        break;
112
113        default:
114            break;
115    }
116 }
117
118 void CsvParser::printSingleData(const strSingleData &data) const {
119     cout << "Mode-" << data.iModbusType << " ";
120     cout << "Slave Number-" << data.uiSlaveNo << " ";
121     cout << "Function-" << data.iFc << " ";
122     cout << "Address-" << data.uiAddress << " ";
123     cout << "Number of registers-" << data.iQtyRegs << " ";
124     cout << "OpcUa index-" << data.iOpcUaIndex << endl;
125 }
126
127 void CsvParser::printPlant(const arrPlant &planta) const {
128     for (int i = 0; i < 4; i++) {
129         for (int j = 0; j < 5; j++) {
130             printSingleData(planta[i][j]);
131         }
132         cout << endl;
133     }
134 }
135
136 } // end of namespace Parser
137

```