

CSV Handler [MACRO LIB]

Author : csl

E-Mail : 3079625093@qq.com

OverView

This is a library implemented with C + + macros to read and write CSV files. It is simple and universal.

Macros

- CSV_READ_FILE(fileName, splitor, itemType, ...)
- CSV_READ_FILE_H(fileName, splitor, itemType, ...)
- CSV_READ_IFS_ALL(ifstream, splitor, itemType, ...)
- CSV_READ_IFS_ALL_H(ifstream, splitor, itemType, ...)
- CSV_READ_IFS_CER(ifstream, splitor, itemNum, itemType, ...)
- CSV_READ_IFS_CER_H(ifstream, splitor, itemNum, itemType, ...)
- CSV_HEADER(...)
- CSV_WRITE_OFS(ofstream, data, splitor, itemType, ...)
- CSV_WRITE_OFS_H(ofstream, header, data, splitor, itemType, ...)
- CSV_WRITE_FILE(fileName, data, splitor, itemType, ...)
- CSV_WRITE_FILE_H(fileName, header, data, splitor, itemType, ...)

Examples

- CSV_READ_FILE(fileName, splitor, itemType, ...)

```
1  /**
2   * @brief read all items in the ifstream
3   *
4   * @param ifstream the input fstream
5   * @param splitor the splitor
6   * @param itemType the type of the item in the csv file
7   * @param ... the types of the members,
8   *           it's order is same as the declaration sequence of member variables.
9   *
10  * @return std::vector<itemType> data
11  */
12
13 void csv_read_file()
14 {
15     /**
```

```

16         * @brief use macro 'CSV_READ_FILE' to read all items in the file
17         */
18         ns_log::process << "use macro 'CSV_READ_FILE' to read all items in the file" << ns_log::endl;
19         auto info = CSV_READ_FILE("../data/info.csv", ',', Info, uint, std::string, float);
20         for (const auto &elem : info)
21             std::cout << elem._gd << ',' << elem._name << ',' << elem._score << std::endl;
22     }
23

```

■ CSV_READ_FILE_H(fileName, splitor, itemType, ...)

```

1     /**
2     * @brief read all items in the ifstream
3     *
4     * @param ifstream the input fstream
5     * @param splitor the splitor
6     * @param itemType the type of the item in the csv file
7     * @param ... the types of the members,
8     *             it's order is same as the declaration sequence of member variables.
9     *
10    * @return std::pair(std::array<std::string, LabNum>, std::vector<itemType>) {header, data}
11    */
12
13    void csv_read_file_h()
14    {
15        /**
16        * @brief use macro 'CSV_READ_FILE_H' to read all items in the file
17        */
18        ns_log::process << "use macro 'CSV_READ_FILE_H' to read all items from file 'refpoint3f.csv'"
19    << ns_log::endl;
20        auto rps = CSV_READ_FILE_H("../data/refpoint3f.csv", ',', ns_geo::RefPoint3f, uint, float,
21    float, float);
22        ns_log::info << "header : ";
23        for (const auto &label : rps.first)
24            ns_log::info << label << ' ';
25        ns_log::info << ns_log::endl;
26        for (const auto &elem : rps.second)
27            std::cout << elem << std::endl;
28    }
29

```

■ CSV_READ_IFS_ALL(ifstream, splitor, itemType, ...)

```

1     /**
2     * @brief read all items in the ifstream
3     *
4     * @param ifstream the input fstream
5     * @param splitor the splitor
6     * @param itemType the type of the item in the csv file
7     * @param ... the types of the members,
8     *             it's order is same as the declaration sequence of member variables.
9     *
10    * @return std::vector<itemType> data
11    */
12
13    ns_log::process << "use macro 'CSV_READ_IFS_ALL' to read all rest items from file 'info.csv'" <<
14    ns_log::endl;
15    auto rps2 = CSV_READ_IFS_ALL(ifs, ',', Info, uint, std::string, float);
16    for (const auto &elem : rps2)
17        std::cout << elem._gd << ',' << elem._name << ',' << elem._score << std::endl;
18

```

```

17     ifs.close();
18

```

■ CSV_READ_IFS_ALL_H(istream, splitor, itemType, ...)

```

1  /**
2   * @brief read all items in the ifstream
3   *
4   * @param ifstream the input fstream
5   * @param splitor the splitor
6   * @param itemType the type of the item in the csv file
7   * @param ... the types of the members,
8   *           it's order is same as the declaration sequence of member variables.
9   *
10  * @return std::pair(std::array<std::string, LabNum>, std::vector<itemType>) {header, data}
11  */
12
13  std::ifstream ifs1("../data/refpoint3f.csv", std::ios::in);
14  ns_log::process << "use macro 'CSV_READ_IFS_ALL_H' to read all rest items from file
'refpoint3f.csv'" << ns_log::endl;
15  auto rps_1 = CSV_READ_IFS_ALL_H(ifs1, ',', ns_geo::RefPoint3f, uint, float, float, float);
16  ns_log::info << "header : ";
17  for (const auto &label : rps_1.first)
18      ns_log::info << label << ' ';
19  ns_log::info << ns_log::endl;
20  for (const auto &elem : rps_1.second)
21      std::cout << elem << std::endl;
22  ifs1.close();

```

■ CSV_READ_IFS_CER(istream, splitor, itemNum, itemType, ...)

```

1  /**
2   * @brief read all items in the ifstream
3   *
4   * @param ifstream the input fstream
5   * @param splitor the splitor
6   * @param itemType the type of the item in the csv file
7   * @param itemNum the number of the items to read
8   * @param ... the types of the members,
9   *           it's order is same as the declaration sequence of member variables.
10  *
11  * @return std::vector<itemType> data
12  */
13
14  ns_log::process << "use macro 'CSV_READ_IFS_ALL' to read all rest items from file 'info.csv'" <<
ns_log::endl;
15  auto rps2 = CSV_READ_IFS_ALL(ifs, ',', Info, uint, std::string, float);
16  for (const auto &elem : rps2)
17      std::cout << elem._gd << ',' << elem._name << ',' << elem._score << std::endl;
18  ifs.close();

```

■ CSV_READ_IFS_CER_H(istream, splitor, itemNum, itemType, ...)

```

1  /**
2   * @brief read all items in the ifstream
3   *
4   * @param ifstream the input fstream
5   * @param splitor the splitor

```

```

6      * @param itemType the type of the item in the csv file
7      * @param itemNum the number of the items to read
8      * @param ... the types of the members,
9      *           it's order is same as the declaration sequence of member variables.
10     *
11     * @return std::pair(std::array<std::string, LabNum>, std::vector<itemType>) {header, data}
12     */
13
14     std::ifstream ifs2("../data/refpoint3f.csv", std::ios::in);
15     ns_log::process << "use macro 'CSV_READ_IFS_CER_H' to read all rest items from file
'refpoint3f.csv'" << ns_log::endl;
16     auto rps_2 = CSV_READ_IFS_CER_H(ifs2, ',', 4, ns_geo::RefPoint3f, uint, float, float, float);
17     ns_log::info << "header : ";
18     for (const auto &label : rps_2.first)
19         ns_log::info << label << ' ';
20     ns_log::info << ns_log::endl;
21     for (const auto &elem : rps_2.second)
22         std::cout << elem << std::endl;
23     ifs2.close();

```

■ CSV_HEADER(...)

```

1      /**
2      * @brief generate the array of csv header
3      * @param ... the header strings
4      */
5
6     CSV_HEADER("x", "y", "z")
7     std::array<std::string, 3>{"x", "y", "z"}

```

■ CSV_WRITE_OFS(ofstream, data, splitor, itemType, ...)

```

1      /**
2      * @brief write data to a csv file
3      *
4      * @param ofstream the out fstream
5      * @param data the data array
6      * @param splitor the splitor
7      * @param itemType the type of item
8      * @param ... the [methods | member name] to get members from a item
9      *
10     * @return void
11     */
12
13     void csv_write_ofs()
14     {
15         /**
16         * @brief gen random point2f set
17         */
18         auto ps = ns_geo::PointSet3f::randomGenerator(10, 0.0f, 1.0f, 0.0f, 1.0f, 0.0f, 1.0f);
19
20         /**
21         * @brief use macro 'CSV_WRITE_OFS' to write items
22         */
23         std::ofstream ofs("../data/point3f.csv");
24         ns_log::process << "use macro 'CSV_WRITE_OFS' to write items to file 'point3f.csv'" <<
ns_log::endl;
25         CSV_WRITE_OFS(ofs, ps, ',', ns_geo::Point3f, x(), y(), z());
26         ofs.close();

```

```
27 | }
```

■ CSV_WRITE_OFS_H(ofstream, header, data, splitor, itemType, ...)

```
1  /**
2   * @brief write data to a csv file
3   *
4   * @param ofstream the out fstream
5   * @param data the data array
6   * @param header the header labels
7   * @param splitor the splitor
8   * @param itemType the type of item
9   * @param ... the [methods | member name] to get members from a item
10  *
11  * @return void
12  */
13
14  void csv_write_ofs_h()
15  {
16      /**
17       * @brief gen random point2f set
18       */
19      auto ps = ns_geo::PointSet3f::randomGenerator(10, 0.0f, 1.0f, 0.0f, 1.0f, 0.0f, 1.0f);
20
21      /**
22       * @brief use macro 'CSV_WRITE_OFS' to write items
23       */
24      std::ofstream ofs("../data/point3f_h.csv");
25      ns_log::process << "use macro 'CSV_WRITE_OFS_H' to write header and items to file
'point3f_h.csv'" << ns_log::endl;
26      CSV_WRITE_OFS_H(ofs, CSV_HEADER("x", "y", "z"), ps, ',', ns_geo::Point3f, x(), y(), z());
27      ofs.close();
28  }
```

■ CSV_WRITE_FILE(fileName, data, splitor, itemType, ...)

```
1  /**
2   * @brief write data to a csv file
3   *
4   * @param fileName the file name
5   * @param data the data array
6   * @param splitor the splitor
7   * @param itemType the type of item
8   * @param ... the [methods | member name] to get members from a item
9   *
10  * @return void
11  */
12
13  void csv_write_file()
14  {
15      /**
16       * @brief gen random point2f set
17       */
18      auto ps = ns_geo::PointSet3f::randomGenerator(10, 0.0f, 1.0f, 0.0f, 1.0f, 0.0f, 1.0f);
19
20      /**
21       * @brief use macro 'CSV_WRITE_FILE' to write items
22       */
```

```

23         ns_log::process << "use macro 'CSV_WRITE_FILE' to write items to file 'point3f.csv'" <<
ns_log::endl;
24         CSV_WRITE_FILE("../data/point3f.csv", ps, ',', ns_geo::Point3f, x(), y(), z());
25     }

```

■ CSV_WRITE_FILE_H(fileName, header, data, splitor, itemType, ...)

```

1      /**
2      * @brief write data to a csv file
3      *
4      * @param fileName the file name
5      * @param header the header labels
6      * @param data the data array
7      * @param splitor the splitor
8      * @param itemType the type of item
9      * @param ... the [methods | member name] to get members from a item
10     */
11     * @return void
12     */
13
14     void csv_write_file_h()
15     {
16         /**
17         * @brief gen random point2f set
18         */
19         auto ps = ns_geo::PointSet3f::randomGenerator(10, 0.0f, 1.0f, 0.0f, 1.0f, 0.0f, 1.0f);
20
21         /**
22         * @brief use macro 'CSV_WRITE_FILE' to write items
23         */
24         ns_log::process << "use macro 'CSV_WRITE_FILE_H' to write herader and items to file
'point3f_h.csv'" << ns_log::endl;
25         CSV_WRITE_FILE_H("../data/point3f_h.csv", CSV_HEADER("x", "y", "z"), ps, ',',
ns_geo::Point3f, x(), y(), z());
26     }

```