

# CSV Handler [MACRO LIB]

Author : csl

E-Mail : [3079625093@qq.com](mailto:3079625093@qq.com)

## Overview

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

## Macros

- *CSVReader[IFS]*
- *CSVReader[FILE]*
- *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, ...)*
- *CSVWriter[OFS]*
- *CSVWriter[FILE]*
- *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

- *CSVReader[IFS]*

```

1      std::ifstream ifs("../data/info.csv");
2      ns_csv::CSVReader readerIFS(ifs);
3
4      while (readerIFS.hasNext())
5      {
6          auto items = readerIFS.next();
7          Info obj(std::stoi(items.at(0)), items.at(1), std::stof(items.at(2)));
8          std::cout << obj << std::endl;
9      }
10
11     ifs.close();

```

#### ■ CSVReader[FILE]

```

1      ns_log::process << "using 'CSVReader[fileName]' to read items in the file 'info.csv'" <<
ns_log::endl;
2
3      ns_csv::CSVReader reader("../data/info.csv");
4
5      while (reader.hasNext())
6      {
7          auto items = reader.next();
8          Info obj(std::stoi(items.at(0)), items.at(1), std::stof(items.at(2)));
9          std::cout << obj << std::endl;
10     }
11     ns_log::process << "using 'CSVReader[ifstream]' to read items in the file 'info.csv'" <<
ns_log::endl;
12

```

#### ■ CSV\_READ\_FILE(fileName, splitor, itemType, ...)

```

1      /**
2       * @brief read all items in the ifstream
3       *
4       * @param fileName the file name
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
20         auto info = CSV_READ_FILE("../data/info.csv", ',', Info, uint, std::string, float);
21
22         for (const auto &elem : info)
23             std::cout << elem << std::endl;
24     }
25

```

#### ■ CSV\_READ\_FILE\_H(fileName, splitor, itemType, ...)

```

1  /**
2   * @brief read all items in the ifstream
3   *
4   * @param fileName the file name
5   * @param splitter the splitter
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
21      auto rps = CSV_READ_FILE_H("../data/refpoint3f.csv", ',', ns_geo::RefPoint3f, uint, float,
22  float, float);
23
24      std::cout << "header : ";
25      for (const auto &label : rps.first)
26          ns_log::info << label << ' ';
27      std::cout << std::endl;
28
29      for (const auto &elem : rps.second)
30          std::cout << elem << std::endl;
31  }

```

#### ■ CSV\_READ\_IFS\_ALL(ifstream, splitter, itemType, ...)

```

1  /**
2   * @brief read all items in the ifstream
3   *
4   * @param ifstream the input fstream
5   * @param splitter the splitter
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
14  /**
15   * @brief use macro 'CSV_READ_IFS_ALL' to read all rest items
16   */
17  ns_log::process << "use macro 'CSV_READ_IFS_ALL' to read all rest items from file 'info.csv'" <<
18  ns_log::endl;
19
20  auto rps2 = CSV_READ_IFS_ALL(ifs, ',', Info, uint, std::string, float);
21
22  for (const auto &elem : rps2)
23      std::cout << elem << std::endl;
24
25  ifs.close();

```

#### ■ CSV\_READ\_IFS\_ALL\_H(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::pair(std::array<std::string, LabNum>, std::vector<itemType>) {header, data}
11  */
12
13  ns_log::process << "use macro 'CSV_READ_IFS_ALL_H' to read all rest items from file
14  'refpoint3f.csv'" << ns_log::endl;
15
16  std::ifstream ifs1("../data/refpoint3f.csv", std::ios::in);
17  auto rps_1 = CSV_READ_IFS_ALL_H(ifs1, ',', ns_geo::RefPoint3f, uint, float, float, float);
18
19  std::cout << "header : ";
20  for (const auto &label : rps_1.first)
21      ns_log::info << label << ' ';
22  std::cout << std::endl;
23
24  for (const auto &elem : rps_1.second)
25      std::cout << elem << std::endl;
26
27  ifs1.close();

```

#### ■ CSV\_READ\_IFS\_CER(ifstream, 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  /**
15   * @brief use macro 'CSV_READ_IFS_CER' to read certain items
16   */
17  ns_log::process << "use macro 'CSV_READ_IFS_CER' to read certain items from file 'info.csv'" <<
18  ns_log::endl;
19
20  std::ifstream ifs("../data/info.csv");
21  auto rps1 = CSV_READ_IFS_CER(ifs, ',', 5, Info, uint, std::string, float);
22
23  for (const auto &elem : rps1)
24      std::cout << elem << std::endl;
25
26

```

#### ■ CSV\_READ\_IFS\_CER\_H(ifstream, splitor, itemNum, itemType, ...)

```

1  /**
2   * @brief read all items in the ifstream
3   *
4   * @param ifstream the input fstream
5   * @param splitter the splitter
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  ns_log::process << "use macro 'CSV_READ_IFS_CER_H' to read all rest items from file
'refpoint3f.csv'" << ns_log::endl;
15
16  std::ifstream ifs2("../data/refpoint3f.csv", std::ios::in);
17  auto rps_2 = CSV_READ_IFS_CER_H(ifs2, ',', 4, ns_geo::RefPoint3f, uint, float, float, float);
18
19  std::cout << "header : ";
20  for (const auto &label : rps_2.first)
21      ns_log::info << label << ' ';
22  std::cout << std::endl;
23
24  for (const auto &elem : rps_2.second)
25      std::cout << elem << std::endl;
26
27  ifs2.close();

```

#### ■ CSVWriter[OFS]

```

1  ns_log::process << "using 'CSVWriter[ofstream]' to write items to the file 'point3f.csv'" <<
ns_log::endl;
2
3  std::ofstream ofs("../data/point3f.csv");
4  ns_csv::CSVWriter writerOFS(ofs);
5
6  for (const auto &p : ps)
7      writerOFS.writeItems(',', p.x(), p.y(), p.z());
8
9  ofs.close();

```

#### ■ CSVWriter[FILE]

```

1  ns_log::process << "using 'CSVWriter[fileName]' to write items to the file 'point3f.csv'" <<
ns_log::endl;
2
3  auto ps = ns_geo::PointSet3f::randomGenerator(10, 0.0f, 1.0f, 0.0f, 1.0f, 0.0f, 1.0f);
4  ns_csv::CSVWriter writer("../data/point3f.csv");
5
6  for (const auto &p : ps)
7      writer.writeItems(',', p.x(), p.y(), p.z());

```

#### ■ 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 use macro 'CSV_WRITE_OFS' to write items
17       */
18      ns_log::process << "use macro 'CSV_WRITE_OFS' to write items to file 'point3f.csv'" <<
ns_log::endl;
19
20      /**
21       * @brief gen random point2f set
22       */
23      auto ps = ns_geo::PointSet3f::randomGenerator(10, 0.0f, 1.0f, 0.0f, 1.0f, 0.0f, 1.0f);
24      std::ofstream ofs("../data/point3f.csv");
25
26      CSV_WRITE_OFS(ofs, ps, ',', ns_geo::Point3f, x(), y(), z());
27
28      ofs.close();
29  }

```

#### ■ 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 use macro 'CSV_WRITE_OFS' to write items
18     */
19     ns_log::process << "use macro 'CSV_WRITE_OFS_H' to write header and items to file
'point3f_h.csv'" << ns_log::endl;
20
21     std::ofstream ofs("../data/point3f_h.csv");
22     /**
23     * @brief gen random point2f set
24     */
25     auto ps = ns_geo::PointSet3f::randomGenerator(10, 0.0f, 1.0f, 0.0f, 1.0f, 0.0f, 1.0f);
26
27     CSV_WRITE_OFS_H(ofs, CSV_HEADER("x", "y", "z"), ps, ',', ns_geo::Point3f, x(), y(), z());
28
29     ofs.close();
30 }

```

#### ■ **CSV\_WRITE\_FILE(fileName, data, splitter, 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 splitter the splitter
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
14 void csv_write_file()
15 {
16     /**
17     * @brief use macro 'CSV_WRITE_FILE' to write items
18     */
19     ns_log::process << "use macro 'CSV_WRITE_FILE' to write items to file 'point3f.csv'" <<
ns_log::endl;
20
21     /**
22     * @brief gen random point2f set
23     */
24     auto ps = ns_geo::PointSet3f::randomGenerator(10, 0.0f, 1.0f, 0.0f, 1.0f, 0.0f, 1.0f);
25
26     CSV_WRITE_FILE("../data/point3f.csv", ps, ',', ns_geo::Point3f, x(), y(), z());
27 }

```

#### ■ **CSV\_WRITE\_FILE\_H(fileName, header, data, splitter, 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 splitter the splitter
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 use macro 'CSV_WRITE_FILE' to write items
18     */
19     ns_log::process << "use macro 'CSV_WRITE_FILE_H' to write header and items to file
'point3f_h.csv'" << ns_log::endl;
20
21     /**
22     * @brief gen random point2f set
23     */
24     auto ps = ns_geo::PointSet3f::randomGenerator(10, 0.0f, 1.0f, 0.0f, 1.0f, 0.0f, 1.0f);
25     CSV_WRITE_FILE_H("../data/point3f_h.csv", CSV_HEADER("x", "y", "z"), ps, ',',
ns_geo::Point3f, x(), y(), z());
26 }
```