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.

Macroes

- **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, ...)**

```
/**
 * @brief read all items in the ifstream
 *
 * @param ifstream the input fstream
 * @param splitor the splitor
 * @param itemType the type of the item in the csv file
 * @param ... the types of the members,
 *            it's order is same as the declaration sequence of member variables.
 *
 * @return std::vector<itemType> data
 */

void csv_read_file()
{
    /**
```

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

```

- **CSV_READ_FILE_H(fileName, splitor, itemType, ...)**

```cpp
    /**
     * @brief read all items in the ifstream
     *
     * @param ifstream the input fstream
     * @param splitor the splitor
     * @param itemType the type of the item in the csv file
     * @param ... the types of the members,
     *             it's order is same as the declaration sequence of member variables.
     *
     * @return std::pair(std::array<std::string, LabNum>, std::vector<itemType>) {header, data}
     */

    void csv_read_file_h()
    {
        /**
         * @brief use macro 'CSV_READ_FILE_H' to read all items in the file
         */
        ns_log::process << "use macro 'CSV_READ_FILE_H' to read all items from file 'refpoint3f.csv'" << ns_log::endl;
        auto rps = CSV_READ_FILE_H("../data/refpoint3f.csv", ',', ns_geo::RefPoint3f, uint, float, float, float);
        ns_log::info << "header : ";
        for (const auto &label : rps.first)
            ns_log::info << label << ' ';
        ns_log::info << ns_log::endl;
        for (const auto &elem : rps.second)
            std::cout << elem << std::endl;
    }
```

- **CSV_READ_IFS_ALL(ifstream, splitor, itemType, ...)**

```cpp
    /**
     * @brief read all items in the ifstream
     *
     * @param ifstream the input fstream
     * @param splitor the splitor
     * @param itemType the type of the item in the csv file
     * @param ... the types of the members,
     *             it's order is same as the declaration sequence of member variables.
     *
     * @return std::vector<itemType> data
     */

    ns_log::process << "use macro 'CSV_READ_IFS_ALL' to read all rest items from file 'info.csv'" << ns_log::endl;
    auto rps2 = CSV_READ_IFS_ALL(ifs, ',', Info, uint, std::string, float);
    for (const auto &elem : rps2)
        std::cout << elem._gd << ',' << elem._name << ',' << elem._score << std::endl;
    ifs.close();

```

- **CSV_READ_IFS_ALL_H(ifstream, splitor, itemType, ...)**

```
    /**
     * @brief read all items in the ifstream
     *
     * @param ifstream the input fstream
     * @param splitor the splitor
     * @param itemType the type of the item in the csv file
     * @param ... the types of the members,
     *             it's order is same as the declaration sequence of member variables.
     *
     * @return std::pair(std::array<std::string, LabNum>, std::vector<itemType>) {header, data}
     */

    std::ifstream ifs1("../data/refpoint3f.csv", std::ios::in);
    ns_log::process << "use macro 'CSV_READ_IFS_ALL_H' to read all rest items from file 'refpoint3f.csv'" << ns_log::endl;
    auto rps_1 = CSV_READ_IFS_ALL_H(ifs1, ',', ns_geo::RefPoint3f, uint, float, float, float);
    ns_log::info << "header : ";
    for (const auto &label : rps_1.first)
        ns_log::info << label << ' ';
    ns_log::info << ns_log::endl;
    for (const auto &elem : rps_1.second)
        std::cout << elem << std::endl;
    ifs1.close();
```

- **CSV_READ_IFS_CER(ifstream, splitor, itemNum, itemType, ...)**

```
    /**
     * @brief read all items in the ifstream
     *
     * @param ifstream the input fstream
     * @param splitor the splitor
     * @param itemType the type of the item in the csv file
     * @param itemNum the number of the items to read
     * @param ... the types of the members,
     *             it's order is same as the declaration sequence of member variables.
     *
     * @return std::vector<itemType> data
     */

    ns_log::process << "use macro 'CSV_READ_IFS_ALL' to read all rest items from file 'info.csv'" << ns_log::endl;
    auto rps2 = CSV_READ_IFS_ALL(ifs, ',', Info, uint, std::string, float);
    for (const auto &elem : rps2)
        std::cout << elem._gd << ',' << elem._name << ',' << elem._score << std::endl;
    ifs.close();
```

- **CSV_READ_IFS_CER_H(ifstream, splitor, itemNum, itemType, ...)**

```
    /**
     * @brief read all items in the ifstream
     *
     * @param ifstream the input fstream
     * @param splitor the splitor
     * @param itemType the type of the item in the csv file
     * @param itemNum the number of the items to read
     * @param ... the types of the members,
     *              it's order is same as the declaration sequence of member variables.
     *
     * @return std::pair(std::array<std::string, LabNum>, std::vector<itemType>) {header, data}
     */

    std::ifstream ifs2("../data/refpoint3f.csv", std::ios::in);
    ns_log::process << "use macro 'CSV_READ_IFS_CER_H' to read all rest items from file
'refpoint3f.csv'" << ns_log::endl;
    auto rps_2 = CSV_READ_IFS_CER_H(ifs2, ',', 4, ns_geo::RefPoint3f, uint, float, float, float);
    ns_log::info << "header : ";
    for (const auto &label : rps_2.first)
        ns_log::info << label << ' ';
    ns_log::info << ns_log::endl;
    for (const auto &elem : rps_2.second)
        std::cout << elem << std::endl;
    ifs2.close();
```

- **CSV_HEADER(...)**

```
    /**
     * @brief generate the array of csv header
     * @param ... the header strings
     */

    CSV_HEADER("x", "y", "z")
    std::array<std::string, 3>{"x", "y", "z"}
```

- **CSV_WRITE_OFS(ofstream, data, splitor, itemType, ...)**

```
    /**
     * @brief write data to a csv file
     *
     * @param osftream the out fstream
     * @param data the data array
     * @param splitor the splitor
     * @param itemType the type of item
     * @param ... the [methods | member name] to get members from a item
     *
     * @return void
     */

    void csv_write_ofs()
    {
        /**
         * @brief gen random point2f set
         */
        auto ps = ns_geo::PointSet3f::randomGenerator(10, 0.0f, 1.0f, 0.0f, 1.0f, 0.0f, 1.0f);

        /**
         * @brief use macro 'CSV_WRITE_OFS' to write items
         */
        std::ofstream ofs("../data/point3f.csv");
        ns_log::process << "use macro 'CSV_WRITE_OFS' to write items to file 'point3f.csv'" <<
ns_log::endl;
        CSV_WRITE_OFS(ofs, ps, ',', ns_geo::Point3f, x(), y(), z());
        ofs.close();
    }
```

- **CSV_WRITE_OFS_H(ofstream, header, data, splitor, itemType, ...)**

```
    /**
     * @brief write data to a csv file
     *
     * @param osftream the out fstream
     * @param data the data array
     * @param header the header labels
     * @param splitor the splitor
     * @param itemType the type of item
     * @param ... the [methods | member name] to get members from a item
     *
     * @return void
     */

    void csv_write_ofs_h()
    {
        /**
         * @brief gen random point2f set
         */
        auto ps = ns_geo::PointSet3f::randomGenerator(10, 0.0f, 1.0f, 0.0f, 1.0f, 0.0f, 1.0f);

        /**
         * @brief use macro 'CSV_WRITE_OFS' to write items
         */
        std::ofstream ofs("../data/point3f_h.csv");
        ns_log::process << "use macro 'CSV_WRITE_OFS_H' to write header and items to file 
'point3f_h.csv'" << ns_log::endl;
        CSV_WRITE_OFS_H(ofs, CSV_HEADER("x", "y", "z"), ps, ',', ns_geo::Point3f, x(), y(), z());
        ofs.close();
    }
```

- **CSV_WRITE_FILE(fileName, data, splitor, itemType, ...)**

```
    /**
     * @brief write data to a csv file
     *
     * @param fileName the file name
     * @param data the data array
     * @param splitor the splitor
     * @param itemType the type of item
     * @param ... the [methods | member name] to get members from a item
     *
     * @return void
     */

    void csv_write_file()
    {
        /**
         * @brief gen random point2f set
         */
        auto ps = ns_geo::PointSet3f::randomGenerator(10, 0.0f, 1.0f, 0.0f, 1.0f, 0.0f, 1.0f);

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

- **CSV_WRITE_FILE_H(fileName, header, data, splitor, itemType, ...)**

```
    /**
     * @brief write data to a csv file
     *
     * @param fileName the file name
     * @param header the header labels
     * @param data the data array
     * @param splitor the splitor
     * @param itemType the type of item
     * @param ... the [methods | member name] to get members from a item
     *
     * @return void
     */

    void csv_write_file_h()
    {
        /**
         * @brief gen random point2f set
         */
        auto ps = ns_geo::PointSet3f::randomGenerator(10, 0.0f, 1.0f, 0.0f, 1.0f, 0.0f, 1.0f);

        /**
         * @brief use macro 'CSV_WRITE_FILE' to write items
         */
        ns_log::process << "use macro 'CSV_WRITE_FILE_H' to write herader and items to file 'point3f_h.csv'" << ns_log::endl;
        CSV_WRITE_FILE_H("../data/point3f_h.csv", CSV_HEADER("x", "y", "z"), ps, ',', ns_geo::Point3f, x(), y(), z());
    }
```