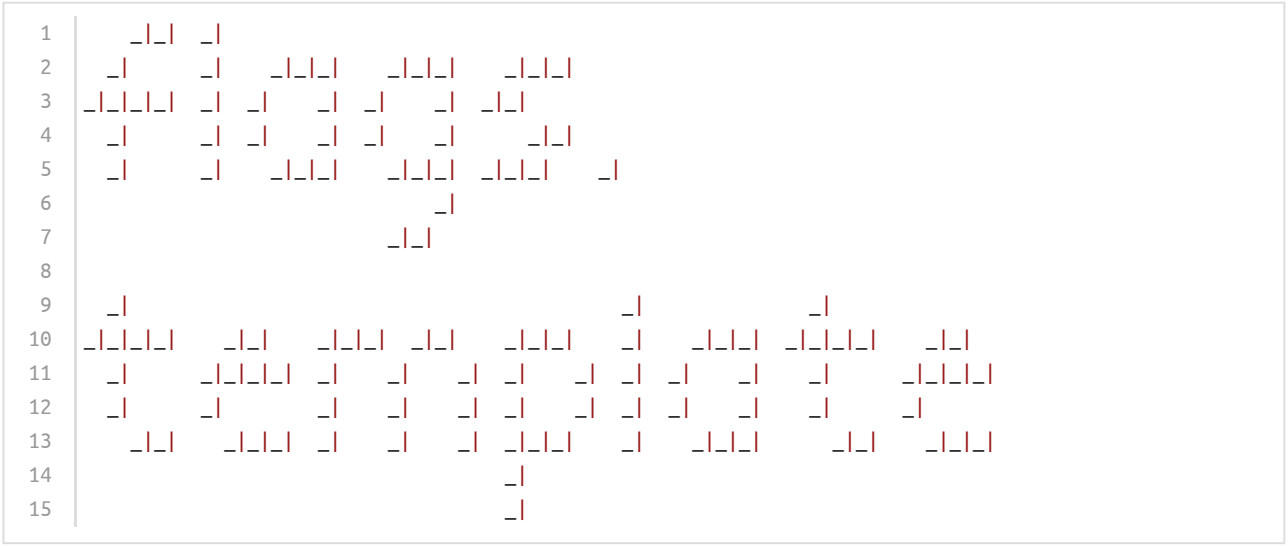
Flags-Template

name: csl

E-Mail: 3079625093@qq.com



OverRide

this is a simple 'program-command-line-parameter-parsing' library using cpp-template.

Usage

```cpp
#include "flags.hpp"

using namespace ns_flags;

int main(int argc, char const* argv[]) {
  /**
   * @brief try-catch is not necessary but it is strongly recommended,
   * because you can get a lot of advice when there are errors in your code
   */
  try {
    ns_flags::ArgParser parser;
    /**
     * @brief define some kinds of arguements
     * [int, std::string, bool, double]
     * std::vector<[int, std::string, bool, double]>
     */
    parser.add_arg<ArgType::INT>("id", 0, "the id of current thread");
    parser.add_arg<ArgType::STRING>("usr", "null", "the name of usr");
    parser.add_arg<ArgType::BOOL>("sex", true,
                                  "the sex of usr [male: true, female: false]");
    parser.add_arg<ArgType::DOUBLE>("height", 1.7, "the height of usr");
```

```cpp
    parser.add_arg<ArgType::INT_VEC>("ids", {1, 2, 3}, "the ids of threads");
    parser.add_arg<ArgType::STRING_VEC>("lans", {"cpp", "python"},
                                        "the used langusges of usr");
    parser.add_arg<ArgType::BOOL_VEC>("choice", {true, false},
                                      "the choice of usr");
    parser.add_arg<ArgType::DOUBLE_VEC>("scores", {2.3, 4.5},
                                        "the score of usr");
    /**
     * @brief set version and help docs
     * @attention if you do not set the help docs, then the help docs
     * will generate automatically
     */
    parser.set_version("2.0");
    // parser.set_help("");

    /**
     * @brief finally, you can set up the parser and then use these arguements
     */
    parser.setup_parser(argc, argv);

    /**
     * @brief print the info of arguements
     */
    for (const auto& [key, value] : parser.get_all_args())
      std::cout << value << std::endl;

    /**
     * @brief use the arguements
     */
    auto id = parser.get_arg_value<ArgType::INT>("id");
    std::cout << "the 'id' I get is: " << id << std::endl;
  } catch (const std::exception& e) {
    std::cerr << e.what() << '\n';
  }
  return 0;
}
```

output

if run command line:

```
./flags --height 98.8 --sex true --usr csl --id 12 --choice true false true --ids 12 34 123 --scores 12.3 45.6 78.9 --lans cpp java python html
```

will output:

```
{'name': choice, 'value': [true, false, true], 'defult': [true, false], 'desc': the choice of usr}
{'name': ids, 'value': [12, 34, 123], 'defult': [1, 2, 3], 'desc': the ids of threads}
{'name': scores, 'value': [12.3, 45.6, 78.9], 'defult': [2.3, 4.5], 'desc': the score of usr}
{'name': lans, 'value': [cpp, java, python, html], 'defult': [cpp, python], 'desc': the used langusges of usr}
{'name': height, 'value': 98.800000, 'defult': 1.700000, 'desc': the height of usr}
{'name': sex, 'value': true, 'defult': true, 'desc': the sex of usr [male: true, female: false]}
{'name': usr, 'value': csl, 'defult': null, 'desc': the name of usr}
{'name': id, 'value': 12, 'defult': 0, 'desc': the id of current thread}
{'name': help, 'value': false, 'defult': false, 'desc': get help docs of this program}
{'name': version, 'value': , 'defult': 1.0, 'desc': the version of this program}
```

if run command line:

```
./flags --help
```

will output:

```
Usage: ./flags [options] [target] ...

    Options        Default Value       Describes
-----------------------------------------------------
  --choice         [true, false]       the choice of usr
  --ids            [1, 2, 3]           the ids of threads
  --scores         [2.3, 4.5]          the score of usr
  --lans           [cpp, python]       the used langusges of usr
  --height         1.700000            the height of usr
  --sex            true                the sex of usr [male: true, female: false]
  --usr            null                the name of usr
  --id             0                   the id of current thread
  --help           false               get help docs of this program
  --version        1.0                 the version of this program
```

if run command line:

```
./flags --version
```

will output:

```
./flags version: 2.0
```

Apis

the types can use

ArgType

```cpp
using INT = int;
using DOUBLE = double;
using BOOL = bool;
using STRING = std::string;
using INT_VEC = std::vector<int>;
using DOUBLE_VEC = std::vector<double>;
using BOOL_VEC = std::vector<bool>;
using STRING_VEC = std::vector<std::string>;
```

each arguement cantains

```cpp
std::string _name;
std::any _value;
std::any _defult_value;
std::string _desc;
```

operate the ArgParser

ArgParser()

```cpp
/**
 * @brief the default and only constructor for ArgParser
 */
```

template void add_arg(const std::string &name, const Type &defult_value, const std::string &desc)

```cpp
/**
 * @brief add a arguement to the parser
 *
 * @tparam Type the type of the arguement
 * @param name the name of the arguement
 * @param defult_value the default value of the arguement
 * @param desc the describe of the arguement
 */
```

auto get_argc() const

```cpp
/**
 * @brief get the count of the arguements in the parser
 *
 * @return auto
 */
```

const ArgInfo &get_arg_info(const std::string &name) const

```
/**
 * @brief Get the arg info object in the parser according to the name
 *
 * @param name the name of the arguement
 * @return const ArgInfo&
 */
```

const auto &get_all_args() const

```
/**
 * @brief Get the all arguements in the parser
 *
 * @return const auto&
 */
```

void setup_parser(int argc, char const *argv[])

```
/**
 * @brief Set the up the parser
 *
 * @param argc the count of the arguements
 * @param argv the values of the arguements
 */
```

void set_help(const std::string &str)

```
/**
 * @brief Set the help docs string for the parser
 *
 * @param str the help str to set
 */
```

void set_version(const std::string &str)

```
/**
 * @brief Set the version of the program
 *
 * @param str the version str
 */
```

template inline Type &get_arg_value(const std::string &name)

```
/**
 * @brief Get the value of an arguement according to name
 *
 * @tparam Type the type of this arguement
 * @param name the name of this arguement
 * @return Type&
 */
```

template inline const Type &get_arg_default_value(const std::string &name)

```
/**
 * @brief Get the default value of an arguement according to name
 *
 * @tparam Type the type of this arguement
 * @param name the name of this arguement
 * @return const Type&
 */
```

const std::string &get_arg_desc(const std::string &name) const

```
/**
 * @brief Get the describe of the arguement named 'name'
 *
 * @param name
 * @return const std::string&
 */
```