Sensalyzer

1.0.0

Создано системой Doxygen 1.9.8

Глава 1

Описание проекта

Sensalyzer - это консольное приложение для анализа данных с датчиков.

Данная программа может читать файлы данных, составленные в соответствии с установленным протоколом информационного взаимодействия, и выводить по ним статистику за определенный временной период.

Для успешной сборки проекта и продуктивной работы с приложением рекомендуется ознакомиться с представленной ниже информацией.

1.1 Сборка проекта

Сборку проекта можно осуществить с помощью утилиты CMake или Make.

1.1.1 Утилита CMake

Для сборки проекта с помощью CMake на компьютере должны быть установлены поддерживаемые утилитой компилятор языка C и система сборки.

Команда сборки проекта с помощью утилиты CMake следующая:

```
cmake -S <path-to-source> -B <path-to-build>
```

<path-to-source> - путь до корневого файла CMakeLists.txt проекта.

<path-to-build> - путь до директории, куда будут записаны результаты сборки.

После успешной сборки исполняемый файл приложения Sensalyzer будет находиться по следующему пути: path-to-build/apps/sensalyzer.

1.1.2 Утилита Make

Для сборки проекта с помощью Make на компьютере должен быть установлен компилятор языка C (рекомендуется GCC).

Команда сборки проекта с помощью утилиты Make следующая:

make -f <path-to-source>

<path-to-source> - путь до файла Makefile в корне проекта.

После успешной сборки исполняемый файл приложения Sensalyzer будет находиться в автоматически созданной директории build/ проекта.

1.2 Руководство пользователя

Программа Sensalyzer может быть запущена в одном из следующих режимов:

sensalyzer или sensalyzer -h - вывод общей информации и завершение работы.

sensalyzer -f <path-to-file> - чтение файла <path-to-file> и вывод общей статистики.

sensalyzer -f <path-to-file> -t <year>[.<month>[.<day>[-<hour>[:<minute>]]]] - чтение файла <path-to-file> и вывод статистики за указанный временной период.

Как можно видеть, приложение Sensalyzer поддерживает следующие ключи:

-h - вывод общей информации о приложении с последующим завершением работы.

-f <path-to-file> - указание анализируемого файла данных, составленного в соответствии с установленным протоколом информационного взаимодействия.

-t <year>[.<month>[.<day>[-<hour>[:<minute>]]]] - указание временного периода для вывода статистики.

1.3 Протокол информационного взаимодействия

Для успешного анализа файла данных он должен соответствовать следующим требованиям:

- Каждая элементарная запись с датчика должна быть расположена на новой строке.
- Каждая элементарная запись с датчика должна обладать следующей структурой:

  ГОД;МЕСЯЦ;ДЕНЬ;ЧАС;МИНУТА;ЗНАЧЕНИЕ_ФИЗИЧЕСКОЙ_ВЕЛИЧИНЫ

Пример файла данных, удовлетворяющего протоколу информационного взаимодействия:

2021;01;16;01;02;-47
2021;01;16;01;03;-44
2021;02;16;01;02;-10
2021;02;16;01;03;-5
2021;03;16;01;02;-3
2021;03;16;01;03;2

В случае отклонения данных от принятого стандарта программа Sensalyzer выведет на экран соответствующее уведомление. При этом в сообщениях об ошибках приняты следующие аббревиатуры.

Аббревиатура	Расшифровка
UTST	"Unknown timestamp" - некорректная временная метка
NSD	"No suitable data" - нет данных, удовлетворяющих заданному временному периоду
NVD	"No valid data" - нет достоверных данных, удовлетворяющих заданному временному периоду

Глава 2

Программные модули

2.1 apps

Модуль консольных приложений.

Функции

- int main (int argc, char *argv[])
  Основная функция приложения Sensalyzer.

2.1.1 Подробное описание

Содержит исходный код приложений проекта с точками входа в виде функций main(..). Перечень реализованных приложений:

- sensalyzer - консольный анализатор данных с датчиков (температуры).

2.1.2 Функции

2.1.2.1 main()

```
int main (
            int argc,
            char * argv[] )
```

Аргументы

in	argc	Количество переданных аргументов командной строки.
in	argv	Массив строк, представляющий переданный набор аргументов.

Возвращаемые значения

0	Приложение завершило работу в штатном режиме.
1	Приложение завершило работу с ошибкой при работе со входным файлом.

2.2 common

Модуль общего назначения.

- typedef int() vector_cmp(const void *, const void *)
  Тип функции сравнения элементов вектора.
- bool initVector (vector *vec, size_t typeSize, size_t capacity)
  Инициализировать вектор.
- void clearVector (vector *vec)
  Очистить вектор.
- void addVectorElement (vector *vec, void *data)
  Добавить элемент в конец вектора.
- void delVectorElement (vector *vec, size_t idx)
  Удалить элемент из вектора.
- void * getVectorElement (const vector *vec, size_t idx)
  Получить элемент вектора.
- bool isVectorEmpty (const vector *vec)
  Проверить, является ли вектор пустым.
- size_t getVectorSize (const vector *vec)
  Получить размер вектора.
- void qsortVector (vector *vec, vector_cmp *comparator)
  Отсортировать вектор.

- #define MIN_YEAR 1900U
  Минимальный номер года.
- #define VOID_YEAR (MIN_YEAR - 1U)
  Незаданный номер года.
- #define MIN_MONTH 1U
  Минимальный номер месяца.
- #define MAX_MONTH 12U
  Максимальный номер месяца.
- #define VOID_MONTH (MAX_MONTH + 1U)
  Незаданный номер месяца.
- #define MIN_DAY 1U
  Минимальный номер дня.
- #define MAX_DAY_28 28U
  Максимальный номер дня во 2 месяце невисокосного года.
- #define MAX_DAY_29 29U
  Максимальный номер дня во 2 месяце високосного года.
- #define MAX_DAY_30 30U
  Максимальный номер дня в 4, 6, 9, 11 месяцах.
- #define MAX_DAY_31 31U

Максимальный номер дня в 1, 3, 5, 7, 8, 10, 12 месяцах.

- #define VOID_DAY (MAX_DAY_31 + 1U)

  Незаданный номер дня.
- #define MIN_HOUR 0U

  Минимальное число часов.
- #define MAX_HOUR 23U

  Максимальное число часов.
- #define VOID_HOUR (MAX_HOUR + 1U)

  Незаданное число часов.
- #define MIN_MINUTE 0U

  Минимальное число минут.
- #define MAX_MINUTE 59U

  Максимальное число минут.
- #define VOID_MINUTE (MAX_MINUTE + 1U)

  Незаданное число минут.
- enum month {
  January = 1 , February , March , April ,
  May , June , July , August ,
  September , October , November , December }

  Названия месяцев.
- bool isTimestampValid (const timestamp *timestamp)

  Определить достоверность временной метки.
- void makeTimestamp (timestamp *timestamp, const char *string)

  Сформировать временную метку по формат-строке.
- void makeVoidTimestamp (timestamp *timestamp)

  Сделать временную метку пустой.
- bool isVoidTimestamp (const timestamp *timestamp)

  Определить, является ли временная метка пустой.
- int cmpTimestamps (const timestamp *first, const timestamp *second)

  Сравнить хронологически две временные метки.
- bool isSubTimestamp (const timestamp *base, const timestamp *sub)

  Проверить, входит ли метка sub во временную область, заданную меткой base.
- void printTimestamp (const timestamp *timestamp)

  Вывести в консоль временную метку.

2.2.1 Подробное описание

Содержит исходный код программных объектов, общеприменимых в рамках проекта:

- vector - контейнер, представляющий собой коллекцию переменного размера произвольного типа данных.
- timestamp - временная метка, позволяющая задать временной период или конкретное время события.

2.2.2 Функции

2.2.2.1 isTimestampValid()

```
bool isTimestampValid (
            const timestamp * timestamp )
```

Аргументы

in	timestamp	Временная метка.

Возвращаемые значения

true	Временная метка достоверна.
false	Временная метка недостоверна.

2.2.2.2 makeTimestamp()

```
void makeTimestamp (
        timestamp * timestamp,
        const char * string )
```

Если формат-строка пустая или содержит недостоверные данные, то временная метка будет сделана пустой.

Аргументы

out	timestamp	Временная метка.
in	string	Строка формата "ГГГГ.ММ.ДД-ЧЧ:ММ", по которой будет сформирована временная метка.

2.2.2.3 makeVoidTimestamp()

```
void makeVoidTimestamp (
        timestamp * timestamp )
```

Аргументы

out	timestamp	Временная метка, которую необходимо сделать пустой.

2.2.2.4 isVoidTimestamp()

```
bool isVoidTimestamp (
        const timestamp * timestamp )
```

Аргументы

in	timestamp	Проверяемая временная метка.

Возвращаемые значения

true	Временная метка является пустой.

Возвращаемые значения

false	Временная метка не является пустой.

2.2.2.5 cmpTimestamps()

```
int cmpTimestamps (
        const timestamp * first,
        const timestamp * second )
```

Аргументы

in	first	Первая временная метка.
in	second	Вторая временная метка.

Возвращаемые значения

Отрицательное	Если first < second.
0	Если first = second.
Положительное	Если first > second.

2.2.2.6 isSubTimestamp()

```
bool isSubTimestamp (
        const timestamp * base,
        const timestamp * sub )
```

Аргументы

in	base	Временная метка, определяющая временную область.
in	sub	Временная метка, вхождение которой необходимо проверить.

Возвращаемые значения

true	Метка sub входит во временную область метки base.
false	Метка sub не входит во временную область метки base.

2.2.2.7 printTimestamp()

```
void printTimestamp (
        const timestamp * timestamp )
```

Выводит только достоверные компоненты временной метки.

Аргументы

in	timestamp	Временная метка для вывода.

2.2.2.8 initVector()

```
bool initVector (
                vector * vec,
                size_t typeSize,
                size_t capacity )
```

Аргументы

out	vec	Инициализируемый вектор.
in	typeSize	Размер хранимого в векторе элемента в байтах.
in	capacity	Начальная вместимость вектора.

Возвращаемые значения

true	Инициализация прошла успешно.
false	Инициализация прошла неудачно.

2.2.2.9 clearVector()

```
void clearVector (
                vector * vec )
```

Аргументы

out	vec	Очищаемый вектор.

2.2.2.10 addVectorElement()

```
void addVectorElement (
                vector * vec,
                void * data )
```

Аргументы

out	vec	Вектор.
in	data	Элемент, добавляемый в конец вектора.

2.2.2.11 delVectorElement()

void delVectorElement (
vector * vec,
size_t idx)

Аргументы

out	vec	Вектор.
in	idx	Индекс удаляемого элемента.

2.2.2.12 getVectorElement()

void * getVectorElement (
const vector * vec,
size_t idx)

Аргументы

in	vec	Вектор.
in	idx	Индекс элемента.

Возвращаемые значения

Элемент	Если индекс имеет допустимое значение.
NULL	Если индекс имеет недопустимое значение.

2.2.2.13 isVectorEmpty()

bool isVectorEmpty (
const vector * vec)

Аргументы

in	vec	Вектор.

Возвращаемые значения

true	Вектор является пустым.
false	Вектор не является пустым.

2.2.2.14 getVectorSize()

size_t getVectorSize (
const vector * vec)

Аргументы

in	vec	Вектор.

Возвращает

Размер вектора.

2.2.2.15 qsortVector()

```
void qsortVector (
            vector * vec,
            vector_cmp * comparator )
```

Для сортировки используется функция qsort() стандартной библиотеки C.

Аргументы

out	vec	Вектор.
in	comparator	Указатель на функцию сравнения элементов вектора.

2.3 sensors

Модуль обработки данных с датчиков.

- #define MIN_TEMPERATURE -99
  Минимальное значение температуры.
- #define MAX_TEMPERATURE 99
  Максимальное значение температуры.
- bool readTempFromFile (const char *path, vector *records)
  Прочитать из файла данные с датчика температуры в соответствии с протоколом информационного взаимодействия.
- void makeTempRecord (temp_record *record, int8_t value)
  Сформировать значение температуры с достоверностью.
- void qsortTempByTimestamp (vector *records)
  Отсортировать массив данных с датчика температуры по временной метке.
- void qsortTempByRecord (vector *records)
  Отсортировать массив данных с датчика температуры по значению температуры.
- void printTempRecord (const temp_record *record)
  Вывести в консоль значение температуры с достоверностью.
- void printPeriodTempStats (const vector *records, const timestamp *timestamp)
  Вывести в консоль статистику по температуре за указанный временной период.
- void printGlobalTempStats (const vector *records)
  Вывести в консоль статистику по температуре за указанные в массиве данных годы и месяцы.

2.3.1 Подробное описание

Содержит исходный код обработчиков данных с датчиков. Перечень реализованных обработчиков данных:

- temperature - обработчик температуры: способен считать данные из csv-файла и вывести по ним статистику.

2.3.2 Функции

2.3.2.1 readTempFromFile()

```
bool readTempFromFile (
        const char * path,
        vector * records )
```

При обнаружении данных, отличных от протокола, выводит в консоль сообщение с указанием строки файла, в которой содержится некорректная информация.

Аргументы

in	path	Путь до файла с данными.
out	records	Массив данных, который будет заполнен информацией из переданного файла.

Возвращаемые значения

true	Чтение данных прошло успешно.
false	Чтение данных завершилось ошибкой.

2.3.2.2 makeTempRecord()

```
void makeTempRecord (
        temp_record * record,
        int8_t value )
```

Если value лежит в допустимом диапазоне температур, то в record будет записано достоверное значение, а иначе - недостоверное.

Аргументы

out	record	Формируемое значение температуры с достоверностью.
in	value	Значение температуры.

2.3.2.3 qsortTempByTimestamp()

```
void qsortTempByTimestamp (
        vector * records )
```

Аргументы

out	records	Массив данных с датчика температуры.

2.3.2.4 qsortTempByRecord()

void qsortTempByRecord (
vector * records)

Аргументы

out	records	Массив данных с датчика температуры.

2.3.2.5 printTempRecord()

void printTempRecord (
const temp_record * record)

Аргументы

in	record	Значение температуры с достоверностью для вывода.

2.3.2.6 printPeriodTempStats()

void printPeriodTempStats (
const vector * records,
const timestamp * timestamp)

Предполагается, что на вход подается отсортированный по временной метке массив данных с датчика температуры. Если данные по температуре за указанный временной период отсутствуют или недостоверны, то выведет в консоль соответствующее сообщение, а иначе - статистику в формате "Timestamp: XXXX.XX.XX-XX:XX | Minimum: XX | Maximum: XX | Average: XX.XXX".

Аргументы

in	records	Массив данных с датчика температуры.
in	timestamp	Временная метка, определяющая временной период, за который необходимо вывести статистику.

2.3.2.7 printGlobalTempStats()

void printGlobalTempStats (
const vector * records)

Предполагается, что на вход подается отсортированный по временной метке массив данных с датчика температуры. Если данные по температуре за какой-либо год или месяц отсутствуют или

недостоверны, то выведет в консоль соответствующее сообщение, а иначе - статистику вида "Год -> Месяцы" в формате "Timestamp: XXXX.XX.XX-XX:XX | Minimum: XX | Maximum: XX | Average: XX.XXX".

Аргументы

in	records	Массив данных с датчика температуры.

2.4 tools

Модуль вспомогательных инструментов.

- void printCmdHelp ()

  Вывести в консоль справку по приложению.
- void handleCmdCommands (int argc, char *argv[], bool *isHelpReceived, char **path, char **timeDate)

  Обработать аргументы командной строки.

2.4.1 Подробное описание

Содержит исходный код вспомогательных инструментов, а именно:

- cmd_handler - обработчик аргументов командной строки.

2.4.2 Функции

2.4.2.1 printCmdHelp()

void printCmdHelp ( )

Справка по приложению представляет собой описание функционала программы, а также список ключей, их назначение и примеры использования.

2.4.2.2 handleCmdCommands()

void handleCmdCommands (
int argc,
char * argv[],
bool * isHelpReceived,
char ** path,
char ** timeDate)

Аргументы

in	argc	Количество переданных аргументов командной строки.
in	argv	Массив строк, представляющий переданный набор аргументов.
out	isHelpReceived	Флаг запроса вывода в консоль справки по приложению.
out	path	Путь до файла с массивом данных с интересующего датчика.
out	timeDate	Временная метка в виде строки для дальнейшей обработки.

Глава 3

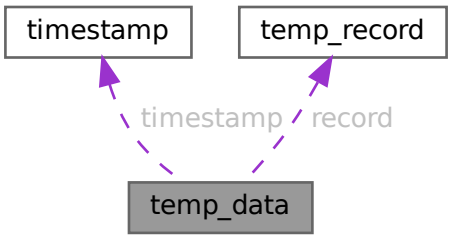
Структуры данных

3.1 Структура temp_data

Данные с датчика температуры.

#include <temperature.h>

Граф связей класса temp_data:



Поля данных

- timestamp timestamp
  Временная метка.
- temp_record record
  Значение.

Объявления и описания членов структуры находятся в файле:

- sensors/temperature.h

3.2 Структура temp_record

Значение температуры с достоверностью.

#include <temperature.h>

Поля данных

- int8_t value
  Значение.
- bool isValid
  Достоверность.

Объявления и описания членов структуры находятся в файле:

- sensors/temperature.h

3.3 Структура timestamp

Временная метка.

#include <timestamp.h>

Поля данных

- uint16_t year
  Год.
- uint8_t month
  Месяц.
- uint8_t day
  День.
- uint8_t hour
  Час.
- uint8_t minute
  Минута.

Объявления и описания членов структуры находятся в файле:

- common/timestamp.h

3.4 Структура vector

Контейнер типа "Вектор".

#include <vector.h>

Поля данных

- void ** data
- size_t typeSize

  Размер элемента данных в байтах.
- size_t size

  Размер массива данных.
- size_t capacity

  Вместимость массива данных.

3.4.1 Подробное описание

Представляет собой усеченный аналог std::vector из C++, т.е. коллекцию переменного размера произвольного типа данных (https://en.cppreference.com/w/cpp/container/vector).

Объявления и описания членов структуры находятся в файле:

- common/vector.h

Предметный указатель