# Students

2.0

Generated by Doxygen 1.12.0

# Chapter 1

# Hierarchical Index

## 1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

# Chapter 2

# Class Index

## 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

# Chapter 3

# File Index

## 3.1 File List

Here is a list of all documented files with brief descriptions:

# Chapter 4

# Class Documentation

## 4.1 Directory_files Struct Reference

**Public Types**

- enum **types** { **Data** , **Results** }

**Public Attributes**

- int **id**
- string **name**
- enum Directory_files::types **type**

The documentation for this struct was generated from the following file:

- include/util.h

## 4.2 File_info Struct Reference

**Public Attributes**

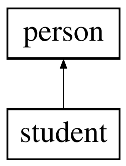
- string **name**
- size_t **size**

The documentation for this struct was generated from the following file:

- include/util.h

## 4.3 person Class Reference

Inheritance diagram for person:



#### Public Member Functions

- virtual std::string **name** () const =0
- virtual std::string **surname** () const =0

#### Protected Member Functions

- **person** (const std::string &name, const std::string surname)

#### Protected Attributes

- std::string **name_**
- std::string **surname_**

The documentation for this class was generated from the following file:

- include/person.h

## 4.4 Record Struct Reference

#### Public Attributes

- double **input** = 0.0
- double **sorting** = 0.0
- double **categorising** = 0.0
- double **output** = 0.0
- double **total** = 0.0
- int **count** = 0

The documentation for this struct was generated from the following file:

- include/util.h

## 4.5 Student Class Reference

A simple student class with basic data variables.

```
#include <student.h>
```

### 4.5.1 Detailed Description

A simple student class with basic data variables.

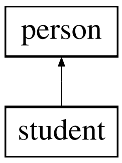
The Student class provides methods to perform student final results calulation and sorting by verious keys.

The documentation for this class was generated from the following file:

- include/student.h

## 4.6 student Class Reference

Inheritance diagram for student:



**Public Member Functions**

- **student** ()

  *Default constructor.*
- **student** (std::string name, std::string surname, std::vector< int > homeworks, int exam)

  *Full constructor.*
- **student** (std::string name, std::string surname)
- **student** (const student &other)

  *Copy constructor;.*
- ∼**student** ()

  *Destructor.*
- void **setName** (const std::string &name)
- void **setSurname** (const std::string &surname)
- void **setHomework** (const std::vector< int > &homework)
- void **setExam** (const int &exam)
- std::string name () const override
- std::string surname () const override
- std::vector< int > **homeworks** () const
- int **exam** () const
- double **final_average** () const
- double **final_median** () const
- student & **operator=** (const student &other)

## Public Member Functions inherited from person

### Private Member Functions

- double final_ave (std::vector< int > homeworks, int exam)

  *Calculates final result by average value.*
- double final_med (std::vector< int > homeworks, int exam)

  *Calculates final result by average value.*

### Private Attributes

- std::vector< int > homeworks_

  *Homework marks.*
- int exam_

  *Exam result.*
- double final_average_

  *Final result using average value of homeworks.*
- double final_median_

  *Final result using median value of homeworks.*

### Friends

- std::ostream & operator<< (std::ostream &os, const student &s)

  *Overlaid method for output operator.*
- std::istream & **operator>>** (std::istream &is, student &s)

### Additional Inherited Members

## Protected Member Functions inherited from person

- **person** (const std::string &name, const std::string surname)

## Protected Attributes inherited from person

- std::string **name_**
- std::string **surname_**

## 4.6.1 Member Function Documentation

### 4.6.1.1 final_ave()

```
double student::final_ave (
              std::vector< int > homeworks,
              int exam)  [private]
```

Calculates final result by average value.

**Parameters**

| | | |
|---|---|---|
| in | *homeworks* | The homework marks. |
| in | *exam* | The exam reuslt. |

**Returns**

The final result.

**Note**

Result formula is average value of homeworks ∗ 0.4 + exam ∗ 0.6.

#### 4.6.1.2 final_med()

```
double student::final_med (
            std::vector< int > homeworks,
            int exam)  [private]
```

Calculates final result by average value.

**Parameters**

| | | |
|---|---|---|
| in | *homeworks* | The homework marks. |
| in | *exam* | The exam reuslt. |

**Returns**

The final result.

**Note**

Result formula is median value of homeworks ∗ 0.4 + exam ∗ 0.6.

#### 4.6.1.3 name()

```
std::string student::name () const  [inline], [override], [virtual]
```

Implements person.

#### 4.6.1.4 surname()

```
std::string student::surname () const  [inline], [override], [virtual]
```

Implements person.

### 4.6.2 Friends And Related Symbol Documentation

#### 4.6.2.1 operator<<

```
std::ostream & operator<< (
            std::ostream & os,
            const student & s)  [friend]
```

Overlaid method for output operator.

**Parameters**

| | | |
|---|---|---|
| `in` | *os* | The out stream. |
| `in` | *s* | Student object. |

Prints students' data in this order: Name, Surname, final_ave, final_med.

**Note**

Uses white spaces (std::setw()) of size 18 and left alligment (std::left).

### 4.6.3 Member Data Documentation

#### 4.6.3.1 exam_

```
int student::exam_  [private]
```

Exam result.

This variable stores the result of the exam.

#### 4.6.3.2 final_average_

```
double student::final_average_  [private]
```

Final result using average value of homeworks.

**Note**

Average value of homeworks $*$ 0.4 + exam $*$ 0.6.

#### 4.6.3.3 final_median_

```
double student::final_median_  [private]
```

Final result using median value of homeworks.

**Note**

Median value of homeworks $*$ 0.4 + exam $*$ 0.6.

#### 4.6.3.4 homeworks_

```
std::vector<int> student::homeworks_  [private]
```

Homework marks.

This vector stores the results of homeworks.

The documentation for this class was generated from the following files:

- include/student.h
- src/student.cpp

## 4.7 Test_data Struct Reference

### Public Attributes

- map< string, Record > **vec_test**
- map< string, Record > **list_test**
- map< string, double > **fg_durations**

The documentation for this struct was generated from the following file:

- include/util.h

## 4.8 Timer Class Reference

### Public Member Functions

- void **reset** ()
- double **elapsed** () const

### Private Types

- using **hrClock** = std::chrono::high_resolution_clock
- using **durationDouble** = std::chrono::duration<double>

### Private Attributes

- std::chrono::time_point< hrClock > **start**

The documentation for this class was generated from the following file:

- include/timer.h

# Chapter 5

# File Documentation

## 5.1 data_util.h

```
#ifndef DATA_UTIL_H_INCLUDED
#define DATA_UTIL_H_INCLUDED

#include "libs.h"
#include "student.h"
#include "util.h"
#include "timer.h"

template<typename Container>
void Input_from_file(Container& local, const string& filename);
template void Input_from_file<vector<student»(vector<student>& local, const string& filename);
template void Input_from_file<list<student»(list<student>& local, const string& filename);

template<typename T>
void output_to_file(T& local, const string& filename, const enum selection& print_by);
template void output_to_file<vector<student»(vector<student>& local, const string& filename, const
enum selection& print_by);
template void output_to_file<list<student»(list<student>& local, const string& filename, const enum
selection& print_by);

template<typename T>
void output_to_screen(T& local);
template void output_to_screen<vector<student»(vector<student>& local);
template void output_to_screen<list<student»(list<student>& local);

template<typename T>
void manual_input(T& container);
template void manual_input<vector<student»(vector<student>& container);
template void manual_input<list<student»(list<student>& container);

void homework_input(vector<int>& homework);

void generate_file(const string& filename, const int& size);

void create_multiple_files(vector<File_info>& files);

void markdown_table();

void test_multiple_files(const vector<string>& files, const enum selection& print_by,
    const string& key, const enum container_types& c_type, const enum strategy& strat);

template<typename T>
void sort_to_categories(T& local, T& Under, T& Over);
template void sort_to_categories<vector<student»(vector<student>& local, vector<student>& Under,
vector<student>& Over);
template void sort_to_categories<list<student»(list<student>& local, list<student>& Under,
list<student>& Over);

template<typename T>
void sort_to_categories2(T& firstc, T& newc);
template void sort_to_categories2<vector<student»(vector<student>& firstc, vector<student>& newc);
template void sort_to_categories2<list<student»(list<student>& firstc, list<student>& newc);

template<typename T>
void sort_to_categories3(T& local, T& over);
template void sort_to_categories3<vector<student»(vector<student>& local, vector<student>& over);
template void sort_to_categories3<list<student»(list<student>& local, list<student>& over);


#endif
```

## 5.2 libs.h

```
#ifndef LIBS_H_INCLUDED
#define LIBS_H_INCLUDED

#include <iostream>
#include <vector>
#include <iomanip>
#include <algorithm>
#include <string>
#include <random>
#include <fstream>
#include <sstream>
#include <numeric>
#include <functional>
#include <map>
#include <list>
#include <type_traits>
#include <ppl.h>
#include <thread>
#include <direct.h>
#include <chrono>

using std::vector;
using std::list;
using std::string;
using std::stringstream;
using std::ifstream;
using std::ofstream;
using std::ios;
using std::map;
using std::function;

using std::numeric_limits;
using std::streamsize;
using std::is_same;

using std::exception;
using std::invalid_argument;
using std::out_of_range;
using std::runtime_error;

using std::random_device;
using std::uniform_int_distribution;

using std::chrono::high_resolution_clock;
using std::chrono::duration;

using std::cout;
using std::cin;
using std::cerr;

using std::log10;
using std::to_string;
using std::pow;
using std::count;
using std::distance;
using std::accumulate;
using std::endl;
using std::setw;
using std::fixed;
using std::setprecision;
using std::left;
using std::sort;
using std::swap;
using std::transform;
using std::stoi;
using std::getline;

#endif
```

## 5.3 person.h

```
#ifndef PERSON_H_INCLUDED
#define PERSON_H_INCLUDED

#include <iostream>
#include <string>

class person {
protected:
    std::string name_;
    std::string surname_;

    person(const std::string& name, const std::string surname)
        : name_(name), surname_(surname) {}

public:

    virtual ~person() = default;

    virtual std::string name() const = 0;
    virtual std::string surname() const = 0;

};

#endif

```

## 5.4 student.h

```
#ifndef STUDENT_H_INCLUDED
#define STUDENT_H_INCLUDED

#include "person.h"
#include <iostream>
#include <vector>
#include <string>
#include <numeric>
#include <algorithm>
#include <random>
#include <list>
#include <map>
#include <type_traits>
#include <ppl.h>
#include <iomanip>
#include <sstream>

class student : public person{

private:

    std::vector<int> homeworks_;

    int exam_;

    double final_average_;

    double final_median_;

    double final_ave(std::vector<int> homeworks, int exam);

    double final_med(std::vector<int> homeworks, int exam);
public:

    friend std::ostream& operator«(std::ostream& os, const student& s);

    friend std::istream& operator»(std::istream& is, student& s);

    student();

    student(std::string name, std::string surname, std::vector<int> homeworks, int exam);

    student(std::string name, std::string surname);

    student(const student& other);

    ~student();


    inline void setName(const std::string& name) { name_ = name; }

    inline void setSurname(const std::string& surname) { surname_ = surname; }

    inline void setHomework(const std::vector<int>& homework) { homeworks_ = homework; }

    inline void setExam(const int& exam) { exam_ = exam; }

    inline std::string name() const override { return name_; }

    inline std::string surname() const  override { return surname_; }

    inline std::vector<int> homeworks() const { return homeworks_; }

    inline int exam() const { return exam_; }

    inline double final_average() const { return final_average_; }

    inline double final_median() const { return final_median_; }


    student& operator=(const student& other);

};

/*
    COMPARATORS
*/

int nam_sur(const student& a, const student& b);

int nam_ave(const student& a, const student& b);

int nam_med(const student& a, const student& b);

int sur_nam(const student& a, const student& b);

int sur_ave(const student& a, const student& b);

int sur_med(const student& a, const student& b);

int ave_nam(const student& a, const student& b);

int ave_sur(const student& a, const student& b);

int ave_med(const student& a, const student& b);

int med_nam(const student& a, const student& b);

int med_sur(const student& a, const student& b);

int med_ave(const student& a, const student& b);

int nam(const student& a, const student& b);

int sur(const student& a, const student& b);

int ave(const student& a, const student& b);

int med(const student& a, const student& b);

template<typename T>
void sort_students(T& Students, const std::string& key);
template void sort_students<std::vector<student»(std::vector<student>& Students, const std::string&
key);
template void sort_students<std::list<student»(std::list<student>& Students, const std::string& key);

#endif
```

## 5.5 timer.h

```
#ifndef TIMER_H_INCLUDED
#define TIMER_H_INCLUDED

#include <iostream>
#include <chrono>
#include <vector>

class Timer {
    using hrClock = std::chrono::high_resolution_clock;
    using durationDouble = std::chrono::duration<double>;
private:
    std::chrono::time_point<hrClock> start;
public:
    Timer() : start{ hrClock::now() } {}
    void reset() {
        start = hrClock::now();
    }
    double elapsed() const {
        return durationDouble(hrClock::now() - start).count();
    }
};

#endif
```

## 5.6 util.h

```
#ifndef UTIL_H_INCLUDED
#define UTIL_H_INCLUDED

#include "libs.h"
#include "student.h"
#include "timer.h"

struct File_info {
    string name;
    size_t size;
};

struct Directory_files {
    int id;
    string name;
    enum types { Data, Results } type;
};

struct Record {

    double input = 0.0;
    double sorting = 0.0;
    double categorising = 0.0;
    double output = 0.0;
    double total = 0.0;
    int    count = 0;
};

struct Test_data {
    map<string, Record> vec_test;
    map<string, Record> list_test;
    map<string, double> fg_durations;
};

enum selection {
    Average,
    Median,
    Both
};

enum strategy {
    s1 = 1, s2 = 2, s3 = 3
};

enum container_types {
    Vector,
    List
};

void update_info(stringstream& info, const enum container_types& type);

void update_files(vector<Directory_files>& files);

void table(const vector<Directory_files> files);

bool is_data_file(const string& filename);

void get_type(const enum container_types& type);

bool is_digits(const string& str);

void progress_clock(const size_t& lines);

void find_keys(string& line, const enum selection& print_by, size_t& n_keys, vector<string>& keys);

enum selection print_selection();

string sort_selection(const enum selection& print_by);

void create_file_selection(vector<File_info>& files);

void file_selection(vector<string>& files);

enum strategy cycle_strat(enum strategy& strat);

#endif
```

# Index