# 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**

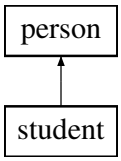
- string **name**
- size_t **size**

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.

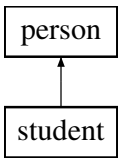
The Student class provides methods to perform student final results calulation and sorting by verious keys.

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

```
00001 #ifndef DATA_UTIL_H_INCLUDED
00002 #define DATA_UTIL_H_INCLUDED
00003
00004 #include "libs.h"
00005 #include "student.h"
00006 #include "util.h"
00007 #include "timer.h"
00008
00009 template<typename Container>
00010 void Input_from_file(Container& local, const string& filename);
00011 template void Input_from_file<vector<student»(vector<student>& local, const string& filename);
00012 template void Input_from_file<list<student»(list<student>& local, const string& filename);
00013
00014 template<typename T>
00015 void output_to_file(T& local, const string& filename, const enum selection& print_by);
00016 template void output_to_file<vector<student»(vector<student>& local, const string& filename, const
      enum selection& print_by);
00017 template void output_to_file<list<student»(list<student>& local, const string& filename, const enum
      selection& print_by);
00018
00019 template<typename T>
00020 void output_to_screen(T& local);
00021 template void output_to_screen<vector<student»(vector<student>& local);
00022 template void output_to_screen<list<student»(list<student>& local);
00023
00024 template<typename T>
00025 void manual_input(T& container);
00026 template void manual_input<vector<student»(vector<student>& container);
00027 template void manual_input<list<student»(list<student>& container);
00028
00029 void homework_input(vector<int>& homework);
00030
00031 void generate_file(const string& filename, const int& size);
00032
00033 void create_multiple_files(vector<File_info>& files);
00034
00035 void markdown_table();
00036
00037 void test_multiple_files(const vector<string>& files, const enum selection& print_by,
00038     const string& key, const enum container_types& c_type, const enum strategy& strat);
00039
00040 template<typename T>
00041 void sort_to_categories(T& local, T& Under, T& Over);
00042 template void sort_to_categories<vector<student»(vector<student>& local, vector<student>& Under,
      vector<student>& Over);
00043 template void sort_to_categories<list<student»(list<student>& local, list<student>& Under,
      list<student>& Over);
00044
00045 template<typename T>
00046 void sort_to_categories2(T& firstc, T& newc);
00047 template void sort_to_categories2<vector<student»(vector<student>& firstc, vector<student>& newc);
00048 template void sort_to_categories2<list<student»(list<student>& firstc, list<student>& newc);
00049
00050 template<typename T>
00051 void sort_to_categories3(T& local, T& over);
00052 template void sort_to_categories3<vector<student»(vector<student>& local, vector<student>& over);
00053 template void sort_to_categories3<list<student»(list<student>& local, list<student>& over);
00054
00055
00056 #endif
```

## 5.2 libs.h

```
00001 #ifndef LIBS_H_INCLUDED
00002 #define LIBS_H_INCLUDED
00003
00004 #include <iostream>
00005 #include <vector>
00006 #include <iomanip>
00007 #include <algorithm>
00008 #include <string>
00009 #include <random>
00010 #include <fstream>
00011 #include <sstream>
00012 #include <numeric>
00013 #include <functional>
00014 #include <map>
00015 #include <list>
00016 #include <type_traits>
00017 #include <ppl.h>
00018 #include <thread>
00019 #include <direct.h>
00020 #include <chrono>
00021
00022 using std::vector;
00023 using std::list;
00024 using std::string;
00025 using std::stringstream;
00026 using std::ifstream;
00027 using std::ofstream;
00028 using std::ios;
00029 using std::map;
00030 using std::function;
00031
00032 using std::numeric_limits;
00033 using std::streamsize;
00034 using std::is_same;
00035
00036 using std::exception;
00037 using std::invalid_argument;
00038 using std::out_of_range;
00039 using std::runtime_error;
00040
00041 using std::random_device;
00042 using std::uniform_int_distribution;
00043
00044 using std::chrono::high_resolution_clock;
00045 using std::chrono::duration;
00046
00047 using std::cout;
00048 using std::cin;
00049 using std::cerr;
00050
00051 using std::log10;
00052 using std::to_string;
00053 using std::pow;
00054 using std::count;
00055 using std::distance;
00056 using std::accumulate;
00057 using std::endl;
00058 using std::setw;
00059 using std::fixed;
00060 using std::setprecision;
00061 using std::left;
00062 using std::sort;
00063 using std::swap;
00064 using std::transform;
00065 using std::stoi;
00066 using std::getline;
00067
00068 #endif
```

## 5.3 person.h

```
00001 #ifndef PERSON_H_INCLUDED
00002 #define PERSON_H_INCLUDED
00003
00004 #include <iostream>
00005 #include <string>
00006
00007 class person {
00008 protected:
00009     std::string name_;
00010     std::string surname_;
```

```
00011
00012     person(const std::string& name, const std::string surname)
00013         : name_(name), surname_(surname) {}
00014
00015 public:
00016
00017     virtual ~person() = default;
00018
00019     virtual std::string name() const = 0;
00020     virtual std::string surname() const = 0;
00021
00022 };
00023
00024 #endif
00025
```

## 5.4 student.h

```
00001 #ifndef STUDENT_H_INCLUDED
00002 #define STUDENT_H_INCLUDED
00003
00004 #include "person.h"
00005 #include <iostream>
00006 #include <vector>
00007 #include <string>
00008 #include <numeric>
00009 #include <algorithm>
00010 #include <random>
00011 #include <list>
00012 #include <map>
00013 #include <type_traits>
00014 #include <ppl.h>
00015 #include <iomanip>
00016 #include <sstream>
00017
00025 class student : public person{
00026
00027 private:
00028
00034     std::vector<int> homeworks_;
00035
00041     int exam_;
00042
00048     double final_average_;
00049
00055     double final_median_;
00056
00066     double final_ave(std::vector<int> homeworks, int exam);
00067
00077     double final_med(std::vector<int> homeworks, int exam);
00078 public:
00079
00090     friend std::ostream& operator<<(std::ostream& os, const student& s);
00091
00095     friend std::istream& operator>>(std::istream& is, student& s);
00096
00102     student();
00103
00107     student(std::string name, std::string surname, std::vector<int> homeworks, int exam);
00108
00112     student(std::string name, std::string surname);
00113
00117     student(const student& other);
00118
00123     ~student();
00124
00125
00129     inline void setName(const std::string& name) { name_ = name; }
00130
00134     inline void setSurname(const std::string& surname) { surname_ = surname; }
00135
00139     inline void setHomework(const std::vector<int>& homework) { homeworks_ = homework; }
00140
00144     inline void setExam(const int& exam) { exam_ = exam; }
00145
00149     inline std::string name() const override { return name_; }
00150
00154     inline std::string surname() const  override { return surname_; }
00155
00159     inline std::vector<int> homeworks() const { return homeworks_; }
00160
00164     inline int exam() const { return exam_; }
00165
```

```
00169     inline double final_average() const { return final_average_; }
00170
00174     inline double final_median() const { return final_median_; }
00175
00176
00180     student& operator=(const student& other);
00181
00182 };
00183
00184 /*
00185     COMPARATORS
00186 */
00187
00188 int nam_sur(const student& a, const student& b);
00189
00190 int nam_ave(const student& a, const student& b);
00191
00192 int nam_med(const student& a, const student& b);
00193
00194 int sur_nam(const student& a, const student& b);
00195
00196 int sur_ave(const student& a, const student& b);
00197
00198 int sur_med(const student& a, const student& b);
00199
00200 int ave_nam(const student& a, const student& b);
00201
00202 int ave_sur(const student& a, const student& b);
00203
00204 int ave_med(const student& a, const student& b);
00205
00206 int med_nam(const student& a, const student& b);
00207
00208 int med_sur(const student& a, const student& b);
00209
00210 int med_ave(const student& a, const student& b);
00211
00212 int nam(const student& a, const student& b);
00213
00214 int sur(const student& a, const student& b);
00215
00216 int ave(const student& a, const student& b);
00217
00218 int med(const student& a, const student& b);
00219
00220 template<typename T>
00221 void sort_students(T& Students, const std::string& key);
00222 template void sort_students<std::vector<student>(std::vector<student>& Students, const std::string&
      key);
00223 template void sort_students<std::list<student>(std::list<student>& Students, const std::string& key);
00224
00225 #endif
```

## 5.5 timer.h

```
00001 #ifndef TIMER_H_INCLUDED
00002 #define TIMER_H_INCLUDED
00003
00004 #include <iostream>
00005 #include <chrono>
00006 #include <vector>
00007
00008 class Timer {
00009     using hrClock = std::chrono::high_resolution_clock;
00010     using durationDouble = std::chrono::duration<double>;
00011 private:
00012     std::chrono::time_point<hrClock> start;
00013 public:
00014     Timer() : start{ hrClock::now() } {}
00015     void reset() {
00016         start = hrClock::now();
00017     }
00018     double elapsed() const {
00019         return durationDouble(hrClock::now() - start).count();
00020     }
00021 };
00022
00023 #endif
```

## 5.6 util.h

```
00001 #ifndef UTIL_H_INCLUDED
00002 #define UTIL_H_INCLUDED
00003
00004 #include "libs.h"
00005 #include "student.h"
00006 #include "timer.h"
00007
00008 struct File_info {
00009     string name;
00010     size_t size;
00011 };
00012
00013 struct Directory_files {
00014     int id;
00015     string name;
00016     enum types { Data, Results } type;
00017 };
00018
00019 struct Record {
00020
00021     double input = 0.0;
00022     double sorting = 0.0;
00023     double categorising = 0.0;
00024     double output = 0.0;
00025     double total = 0.0;
00026     int    count = 0;
00027 };
00028
00029 struct Test_data {
00030     map<string, Record> vec_test;
00031     map<string, Record> list_test;
00032     map<string, double> fg_durations;
00033 };
00034
00035 enum selection {
00036     Average,
00037     Median,
00038     Both
00039 };
00040
00041 enum strategy {
00042     s1 = 1, s2 = 2, s3 = 3
00043 };
00044
00045 enum container_types {
00046     Vector,
00047     List
00048 };
00049
00050 void update_info(stringstream& info, const enum container_types& type);
00051
00052 void update_files(vector<Directory_files>& files);
00053
00054 void table(const vector<Directory_files> files);
00055
00056 bool is_data_file(const string& filename);
00057
00058 void get_type(const enum container_types& type);
00059
00060 bool is_digits(const string& str);
00061
00062 void progress_clock(const size_t& lines);
00063
00064 void find_keys(string& line, const enum selection& print_by, size_t& n_keys, vector<string>& keys);
00065
00066 enum selection print_selection();
00067
00068 string sort_selection(const enum selection& print_by);
00069
00070 void create_file_selection(vector<File_info>& files);
00071
00072 void file_selection(vector<string>& files);
00073
00074 enum strategy cycle_strat(enum strategy& strat);
00075
00076 #endif
```

# Index