# Programme module's description

*Nick Zadubrovsky*

## Contents:



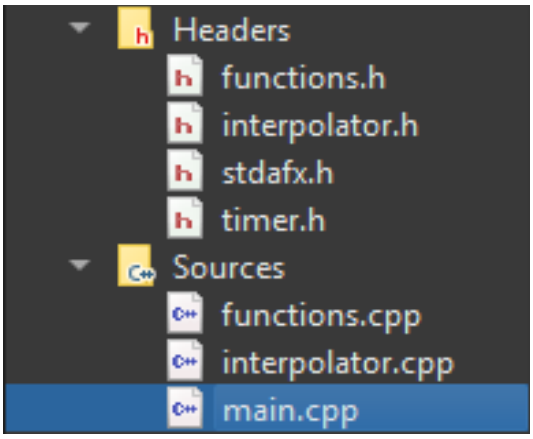
## 1. *Structure.*

Programme's source code is represented by 3 (*.cpp) files each having its own corresponding *header* file (*.h):

- functions.cpp
- interpolator.cpp
- main.cpp

Header files:

- functions.h
- interpolator.h
- stdafx.h
- timer.h




### *functions.cpp*

Contains realisation of methods declared in **functions.h**
In turn, this .cpp is visually split into 6 major groups of functions (just on purpose of an easier navigation through it):

- .TXT FILE PROCESSING FUNCTIONS,
  .INP FILE PROCESSING FUNCTIONS,
  .CSV FILE PROCESSING FUNCTIONS,
  T FIELD FILE PROCESSING FUNCTIONS,
  are responsible for processing data files with an appropriate extensions.
  Each of the above sections is represented by a ***parser function (txtParse, inpParse, csvParse)*** and ***mesh-getter function (GetMeshTXT, GetMeshINP, GetMeshCSV)***. Besides parser and mesh-getter, the CSV section has **GetTempCSV** member function, which purpose is to retrieve T values from a csv table (those are the headers of columns).

```
5
6   /*==============.TXT=FILE=PROCESSING=FUNCTIONS==============================*/
7
8   std::vector<std::vector<double>> txtParse(std::string const& path) {...};
60
61  std::vector<std::pair<double, double>> GetMeshTXT(const std::vector<std::vector<double>>& table, int colNumber){ ...}
72
73  /*==============.INP=FILE=PROCESSING=FUNCTIONS==============================*/
74
75  std::vector<std::vector<double>> inpParse(std::string& path_inp) { ...}
156
157 std::vector<std::pair<double, double>> GetMeshINP(const std::vector<std::vector<double>>& table, double& T) { ...}
223
224 /*==============.CSV=FILE=PROCESSING=FUNCTIONS==============================*/
225
226 std::vector<std::vector<double>> csvParse(std::string const& path, const char delimiter) {...};
291
292 std::vector<std::pair<double, double>> GetMeshCSV(const std::vector<std::vector<double>>& table, double& T){ ...}
358
359 /** ...*/
366 std::vector<double> GetTempCSV(std::string& str){ ...};
397
```

- GRID PROCESSING FUNCTIONS
  includes ***interMesh*** function, which is responsible for interpolation of meshes passed into it.

```
456 /*==============GRID=PROCESSING=FUNCTIONS===================================*/
457
458 std::vector<std::pair<double, double>> interMesh(std::vector<std::pair<double, double>>& m1,
459                                                   std::vector<std::pair<double, double>>& m2) { ...}
481
482 /*==============MATH=FUNCTIONS=AND=BOOLEAN=FILTERS==========================*/
483
484 bool isinX(double x, std::vector<std::pair<double, double>>& mesh) { ...}
494
495 bool isinT(double T, const std::vector<std::vector<double>>& table) { ...}
505
506 double Planck(double lam, double T) {...};
511
512 double integrateTrapezoidal(std::vector<std::pair<double, double>>& F) {...}
523
524 double integrateTrapezoidal(double(&f)(double, double), double min, double max, double N, double T){ ...}
534
535 /*===========================================================================*/
```

- MATH FUNCTIONS AND BOOLEAN FILTERS
  represent mathematical method of integration (***Trapezoidal***, once overloaded), physical functions (***Planck***), and binary predicates (**isinX**, **isinT**).

## ***interpolator.cpp***

Along with its declaration header (***interpolator.h***) represent class which implements linear interpolation (linearly weighted) to mesh passed into its class' object as parameter.
Interpolation by temperatures' grid is conducted implicitly within the procedure of getting mesh out of the data file.
Explicitly this class is only used in ***interMesh*** *function.*

calculation/interpolator.h <Select Symbol>

```cpp
#ifndef INTERPOLATOR_H_
#define INTERPOLATOR_H_

#include <utility>
#include <vector>

class Interpolator {
public:
  //On construction, we take in a vector of data point pairs
  //that represent the line we will use to interpolate
  Interpolator(const std::vector<std::pair<double, double>>&  points);

  //Computes the corresponding Y value
  //for X using linear interpolation
  double findValue(double x) const;

private:
  //Our container of (x,y) data points
  //std::pair::<double, double>.first = x value
  //std::pair::<double, double>.second = y value
  std::vector<std::pair<double, double>> _points;
};
#endif /* INTERPOLATOR_H_ */
```

### *timer.h*

Auxiliary single-filed class used within ***main.cpp*** to estimate physical time of programme's execution. To do the latter, just an instance of the class needs to be declared within the scope of interest (timer starts and ends within the realm enshrouds the instance).

### *stdafx.h*

Contains preprocessor directives with libs to be used further in the programme.

calculation/stdafx.h

```cpp
#ifndef STDAFX_H
#define STDAFX_H

#pragma once

#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <cmath>
#include <vector>
#include <stdlib.h>
#include <algorithm>
#include <iterator>
#include <regex>
#include <thread>
#include "interpolator.h"
#include "timer.h"



using namespace std;


#endif
```

### *main.cpp*

The main file.
Sure enough consists of one function *main.cpp.*