Final Project

Description Document

• Outline

- The final project is just for each **individual** student.
- There are 2 parts, including the **basic** part and the **advanced** part.
- In this project, you will need to report the health data, in terms of overweight/obesity (based on **Body Mass Index**, or, actual weight against **standard weight**), of a certain number of people through calculation.

• Background

- **Body Mass Index (BMI) & Weight Rank**
- BMI is an indicator closely related to the total amount of someone’s body fat and mainly reflects his/her systemic overweight and obesity condition. Since BMI is a percentage of body fat, it is accurate to use BMI, more so than simply his/her weight, to measure the risk of heart disease and high blood pressure due to overweight.
- BMI calculation formula:

$$\mathbf{BMI} = \frac{\boldsymbol{w}}{\boldsymbol{h}^2} \qquad \mathbf{(1)}$$

where $\boldsymbol{w}$ and $\boldsymbol{h}$ are the weight (kg) and height (cm) of the person, respectively.

- Based on the BMI value, the degree of someone’s overweight/obesity can be ranked as in Table 1:

Table 1

Weight Rank	BMI
Underweight	**BMI < 18.5**
Normal range	**18.5 ≤ BMI < 24**
Overweight	**24 ≤ BMI < 27**
Mild obesity	**27 ≤ BMI < 30**
Moderate obesity	**30 ≤ BMI < 35**
Severe obesity	**BMI ≥ 35**

• Background

- **Standard Weight & Overweight Percentage**
- Other than BMI and weight ranks, the overweight percentage is another important indicator to reflect and measure a person's health. A large number of statistical data on different body types show that the standard weight, which can be expressed by the relationship between height and weight, is an ideal and simple indicator reflecting the normal weight of a male/female. Against the standard weight, a person can have his/her overweight percentage evaluated.
- Standard weight calculation formulae (according to the World Health Organization):
    - **Male**: $\boldsymbol{sw} = (\boldsymbol{h} - \mathbf{80}) \times \mathbf{70}\%$ **(2)**
    - **Female**: $\boldsymbol{sw} = (\boldsymbol{h} - \mathbf{70}) \times \mathbf{60}\%$ **(3)**

  where $\boldsymbol{sw}$ is the standard weight (kg), and $\boldsymbol{h}$ is the height (cm) of the person.
- Based on the standard weight, one’s **overweight percentage** can be evaluated as follows:
    - $\mathbf{op} = \frac{\boldsymbol{w}-\boldsymbol{sw}}{\boldsymbol{sw}} \times \mathbf{100}\%$ **(4)**

  where $\boldsymbol{w}$ is the (actual) weight (kg) of the person, and $\boldsymbol{sw}$ is the standard weight (kg) of the person’s gender.

• Basic Part

- Tasks:
  1. Use **arrays** to store the information of a certain number of people, including (at least) their **names**, **genders**, **weights**, and **heights**.
  2. Calculate, store, rank, and sort **BMI** values:
     a) Calculate the value of BMI for each person (using **Equation (1)**), and store the calculated BMI values in an array;
     b) Sort the values of the generated BMI array in the ascending order, and meanwhile record the "sorted indices/subscripts" (which can be used for displaying other information and calculated results like names, overweight percentage, etc, in the same ascending order).
  3. Calculate the value of **overweight percentage** for each person (using **Equations (2)(3)(4)**), and store the calculated values in an array.
  4. Display the information and calculated results, including **names**, **genders**, **BMI values**, **health condition** (in terms of **weight ranks**, which can be determined based on the calculated BMI values as per **Table 1**), and **overweight percentages**, in the terminal **in a sorted, tabular format**.
- Relevant technical points:
  1. Array: 1D arrays, 2D arrays, pointer arrays;
  2. Pointer: pointers to 1D, 2D arrays;
  3. Functions;
  4. Branching;
  5. Looping.

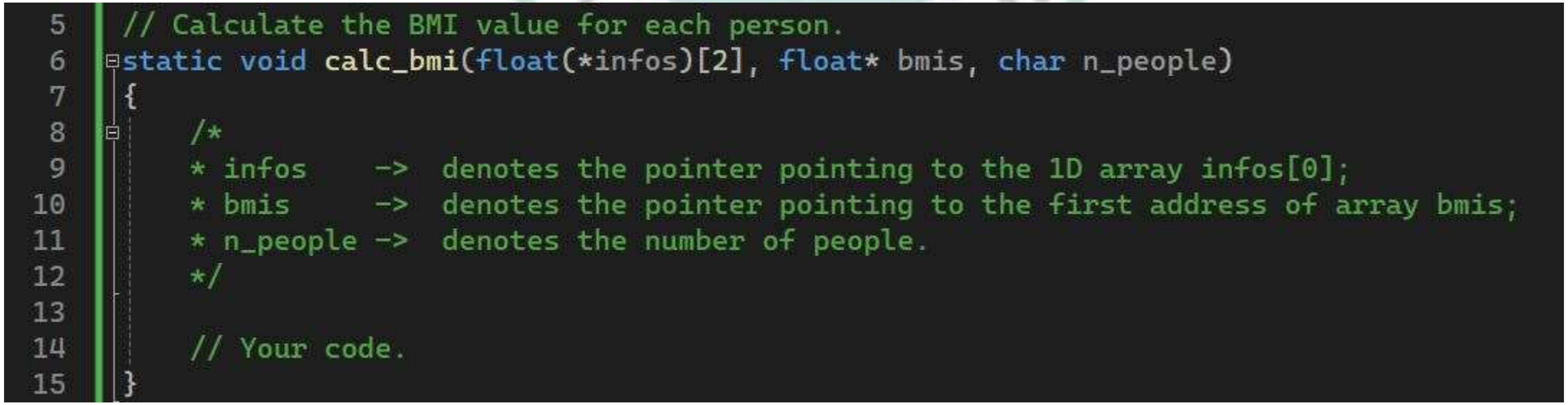
- Basic Part
- Program Structure

- 1, Header files

```
1.#include<stdio.h>
2.#include<math.h>
```

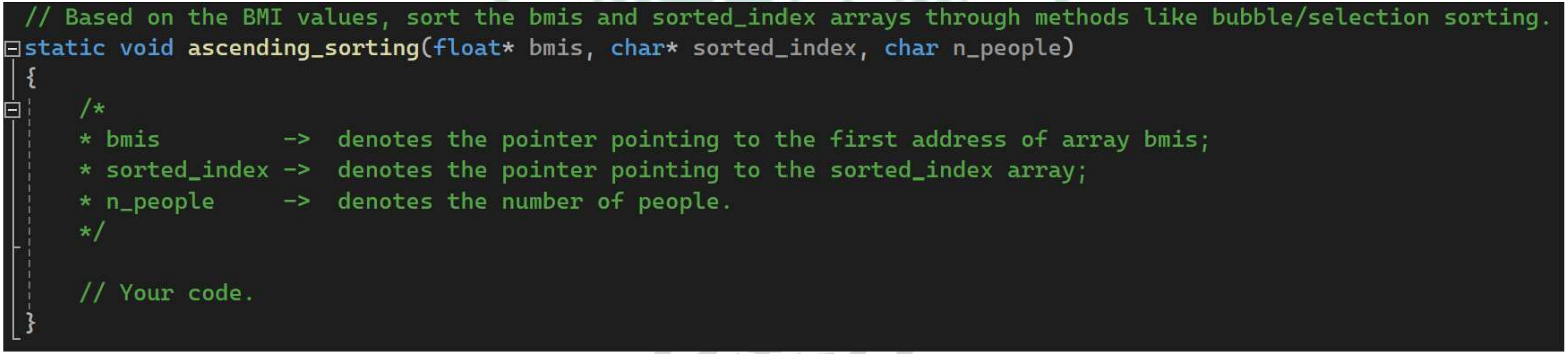
- 2, Functions
- (1), BMI calculator function

You can use function *pow(a, 2)* to calculate the value of *a* squared



```c
// Calculate the BMI value for each person.
static void calc_bmi(float(*infos)[2], float* bmis, char n_people)
{
    /*
    * infos    ->  denotes the pointer pointing to the 1D array infos[0];
    * bmis     ->  denotes the pointer pointing to the first address of array bmis;
    * n_people ->  denotes the number of people.
    */

    // Your code.
}
```

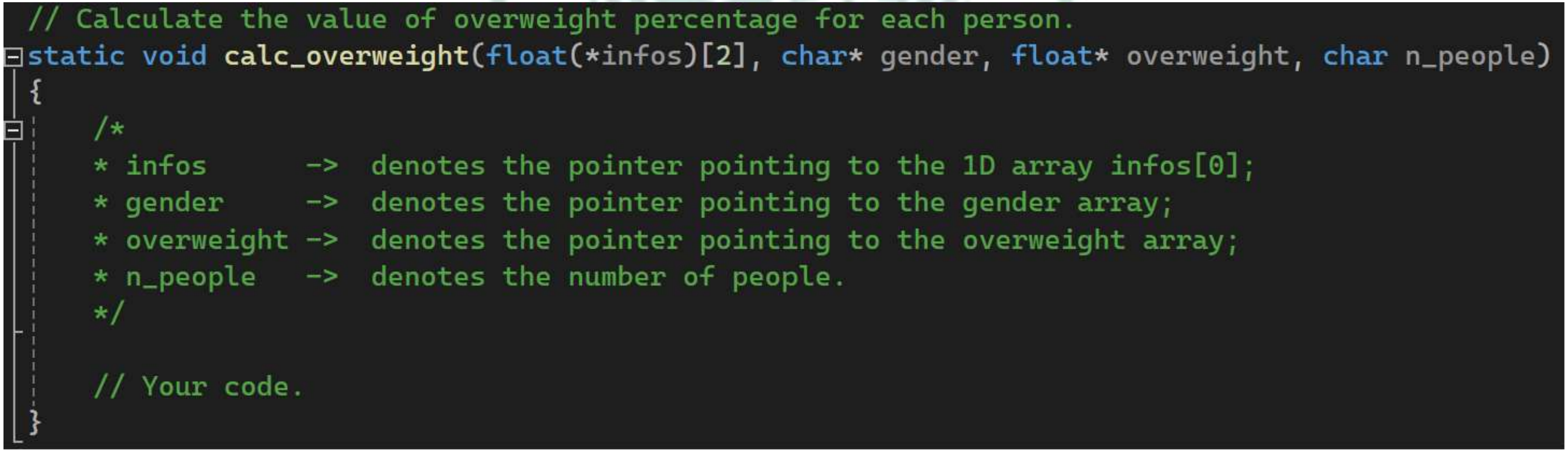
- Basic Part
- Program Structure
- 2, Functions
- (2), BMI sorting function



```
// Based on the BMI values, sort the bmis and sorted_index arrays through methods like bubble/selection sorting.
static void ascending_sorting(float* bmis, char* sorted_index, char n_people)
{
    /*
    * bmis          ->  denotes the pointer pointing to the first address of array bmis;
    * sorted_index  ->  denotes the pointer pointing to the sorted_index array;
    * n_people      ->  denotes the number of people.
    */

    // Your code.
}
```

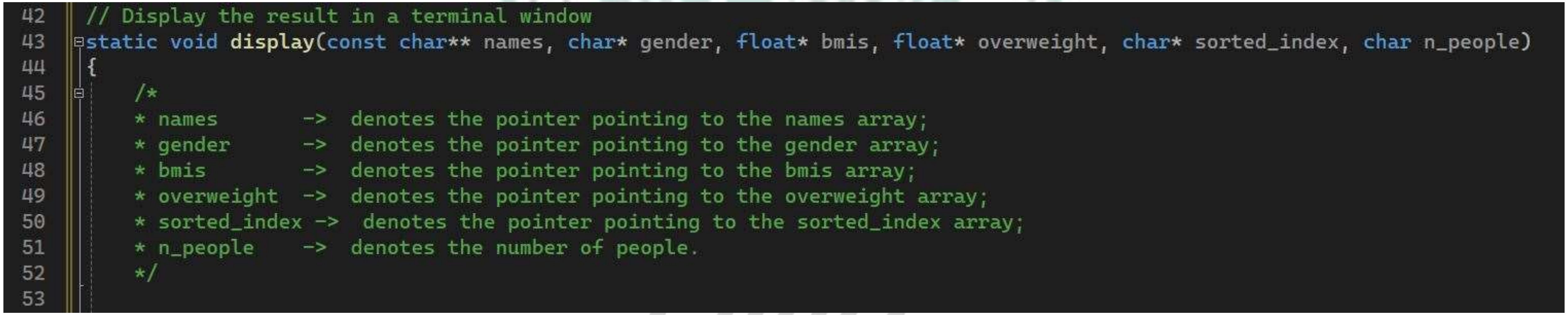
- Basic Part
- Program Structure

- 2, Functions
- (3), overweight percentage calculator function



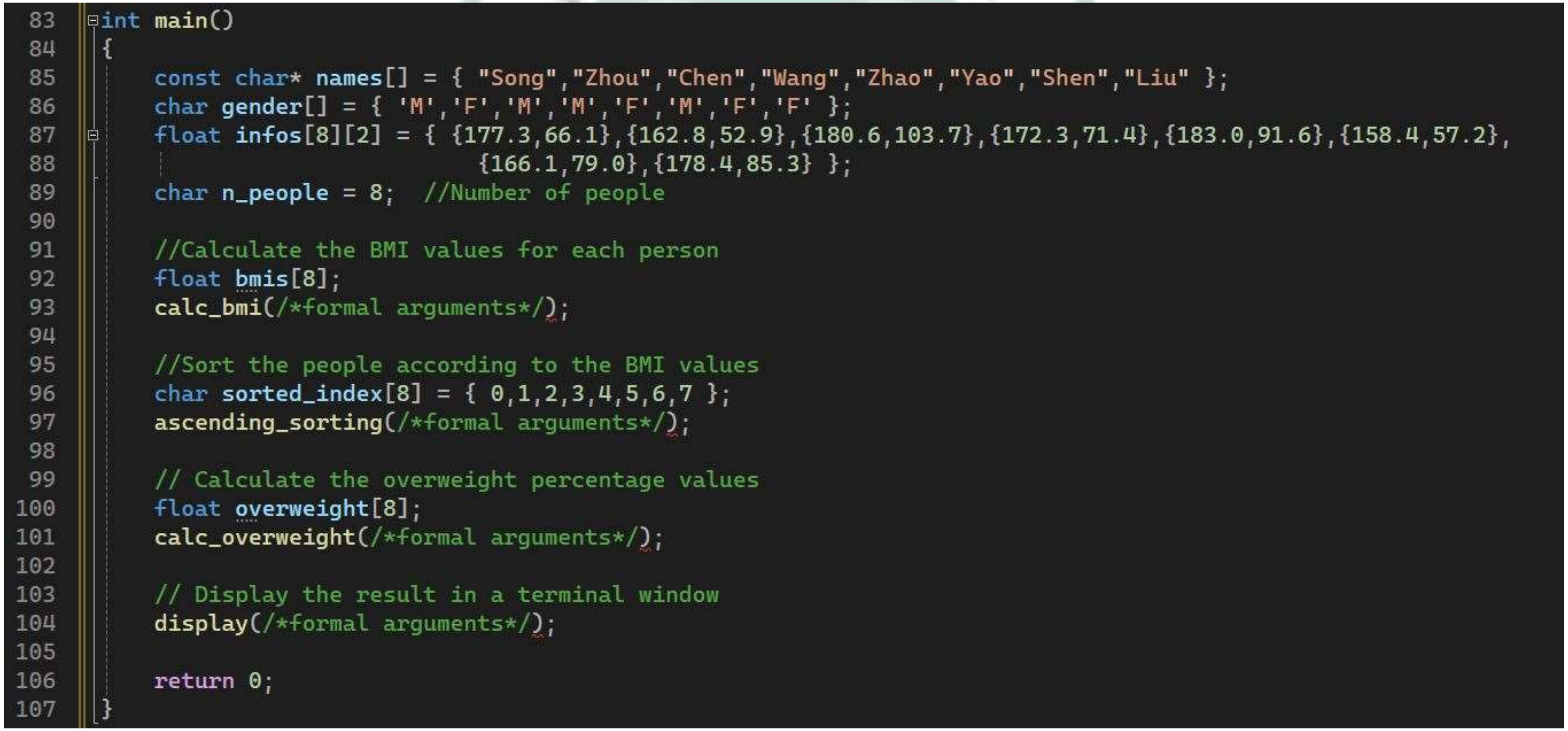
```
// Calculate the value of overweight percentage for each person.
static void calc_overweight(float(*infos)[2], char* gender, float* overweight, char n_people)
{
    /*
    * infos      ->  denotes the pointer pointing to the 1D array infos[0];
    * gender     ->  denotes the pointer pointing to the gender array;
    * overweight ->  denotes the pointer pointing to the overweight array;
    * n_people   ->  denotes the number of people.
    */

    // Your code.
}
```

- Basic Part
- Program Structure
- 2, Functions
- (4), result display function



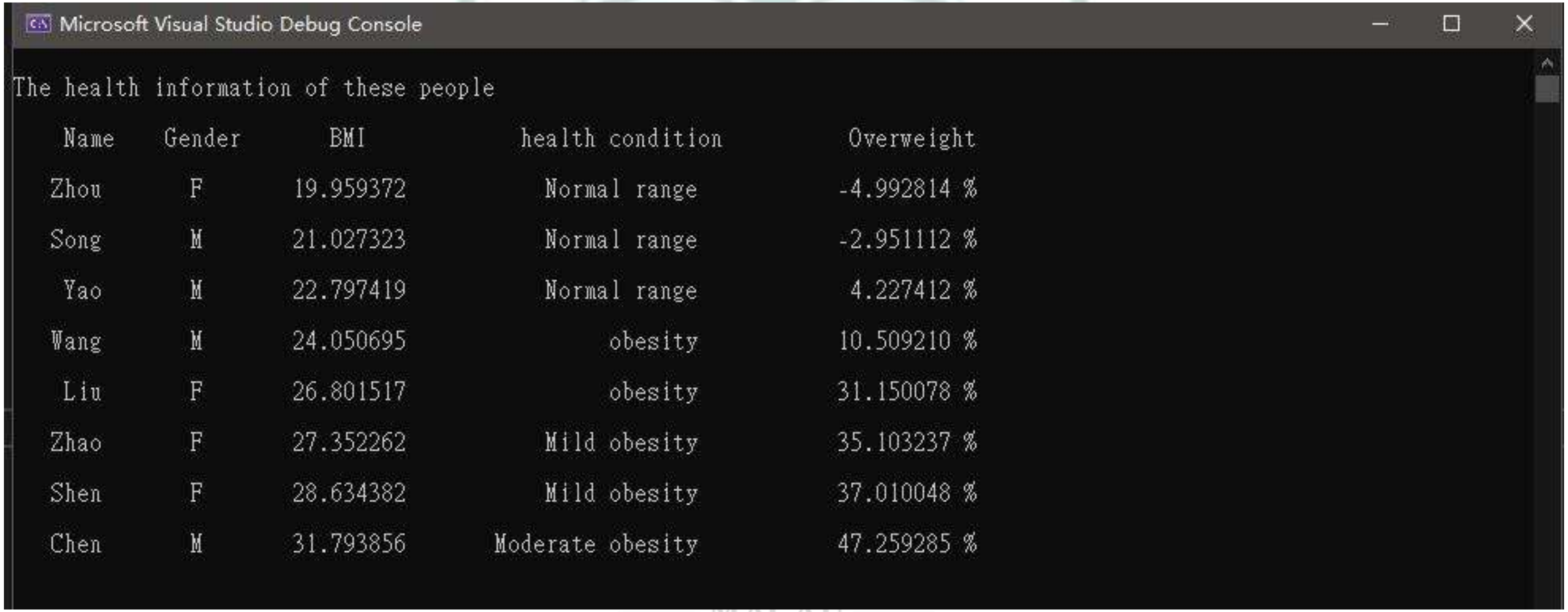
```
// Display the result in a terminal window
static void display(const char** names, char* gender, float* bmis, float* overweight, char* sorted_index, char n_people)
{
    /*
    * names       ->  denotes the pointer pointing to the names array;
    * gender      ->  denotes the pointer pointing to the gender array;
    * bmis        ->  denotes the pointer pointing to the bmis array;
    * overweight  ->  denotes the pointer pointing to the overweight array;
    * sorted_index ->  denotes the pointer pointing to the sorted_index array;
    * n_people    ->  denotes the number of people.
    */

```

- Basic Part
- Program Structure
- 3, Main function



```
int main()
{
    const char* names[] = { "Song","Zhou","Chen","Wang","Zhao","Yao","Shen","Liu" };
    char gender[] = { 'M','F','M','M','F','M','F','F' };
    float infos[8][2] = { {177.3,66.1},{162.8,52.9},{180.6,103.7},{172.3,71.4},{183.0,91.6},{158.4,57.2},
                            {166.1,79.0},{178.4,85.3} };
    char n_people = 8;  //Number of people

    //Calculate the BMI values for each person
    float bmis[8];
    calc_bmi(/*formal arguments*/);

    //Sort the people according to the BMI values
    char sorted_index[8] = { 0,1,2,3,4,5,6,7 };
    ascending_sorting(/*formal arguments*/);

    // Calculate the overweight percentage values
    float overweight[8];
    calc_overweight(/*formal arguments*/);

    // Display the result in a terminal window
    display(/*formal arguments*/);

    return 0;
}
```

- Basic Part
- Result Display (Example)

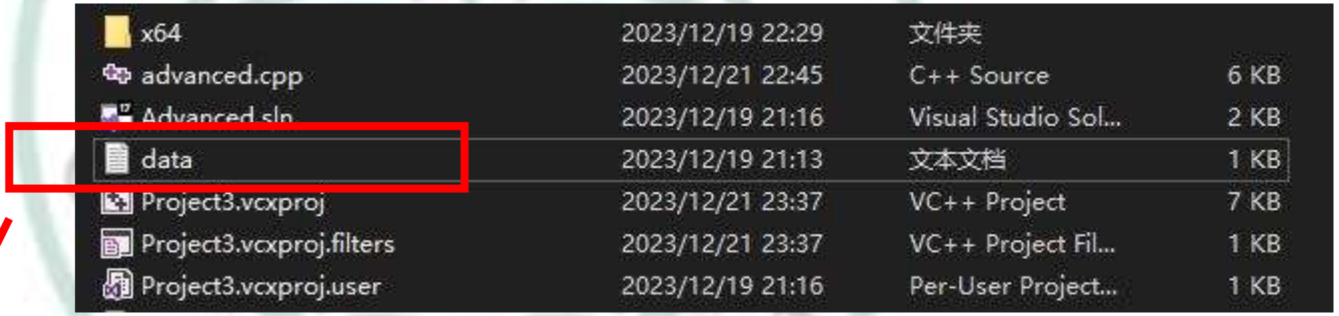


```
The health information of these people
   Name    Gender      BMI           health condition          Overweight
  Zhou       F       19.959372        Normal range           -4.992814 %
  Song       M       21.027323        Normal range           -2.951112 %
   Yao       M       22.797419        Normal range            4.227412 %
  Wang       M       24.050695             obesity           10.509210 %
   Liu       F       26.801517             obesity           31.150078 %
  Zhao       F       27.352262        Mild obesity           35.103237 %
  Shen       F       28.634382        Mild obesity           37.010048 %
  Chen       M       31.793856    Moderate obesity           47.259285 %
```

• Advanced Part

- Tasks:
  1. **Read** the information (i.e., **names**, **genders**, **weights**, and **heights**) of a certain number of people from a data **file**, and use a **structure array** to store these information.
  2. Calculate, store, rank, and sort **BMI** values:
     a) Calculate the value of BMI for each person (using **Equation (1)**), and store the calculated BMI values in the structure array;
     b) Rank the health condition for each person based on the calculated BMI values (using **Table 1**), and store the health conditions (in terms of **weight ranks**) in the structure array;
     c) Sort the values of the generated BMI array in the ascending order, using a pointer array containing pointers to the structure array.
  3. Calculate the value of **overweight percentage** for each person (using **Equations (2)(3)(4)**), and store the calculated values in the structure array.
  4. **Write and save to a file** the information and calculated results, including **names**, **genders**, **BMI values**, **health condition** (in terms of **weight ranks**), and **overweight percentages**, in a sorted, organized format.
- Relevant technical points:
  1. File I/O: reading from or writing to files;
  2. Structure: structure variable definition and structure member access;
  3. Array: 1D arrays, structure arrays, pointer arrays;
  4. Pointer: pointers to 1D and structure arrays;
  5. Functions;
  6. Branching;
  7. Looping.

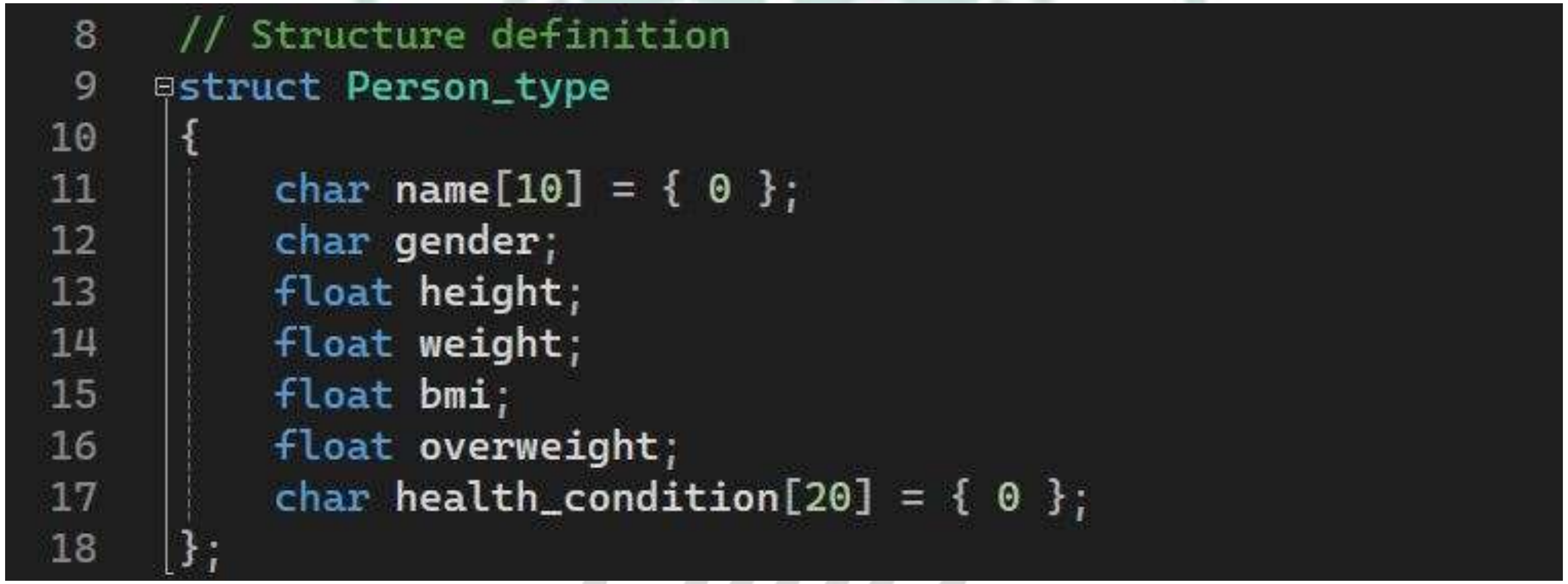
- Advanced Part
- Preparation of the Input File before Program Execution



x64	2023/12/19 22:29	文件夹	
advanced.cpp	2023/12/21 22:45	C++ Source	6 KB
Advanced.sln	2023/12/19 21:16	Visual Studio Sol...	2 KB
data	2023/12/19 21:13	文本文档	1 KB
Project3.vcxproj	2023/12/21 23:37	VC++ Project	7 KB
Project3.vcxproj.filters	2023/12/21 23:37	VC++ Project Fil...	1 KB
Project3.vcxproj.user	2023/12/19 21:16	Per-User Project...	1 KB

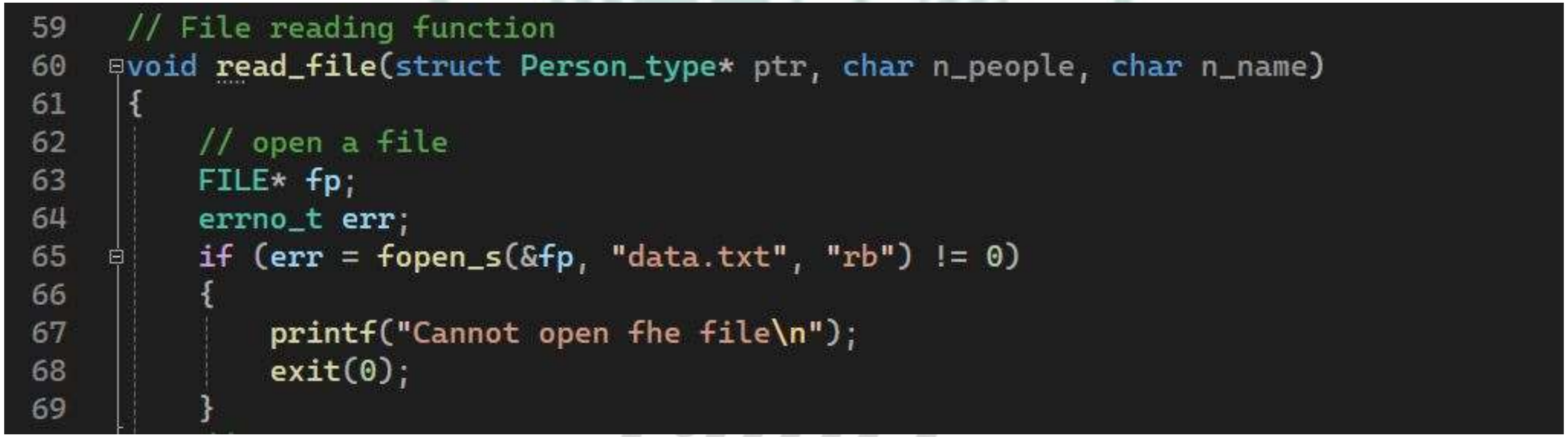
The file “data.txt” is used to input the information and should be placed in the same/root folder of the source code file “advanced.cpp” for reading.

- Advanced Part
- Program Structure
- 1, Structure definition



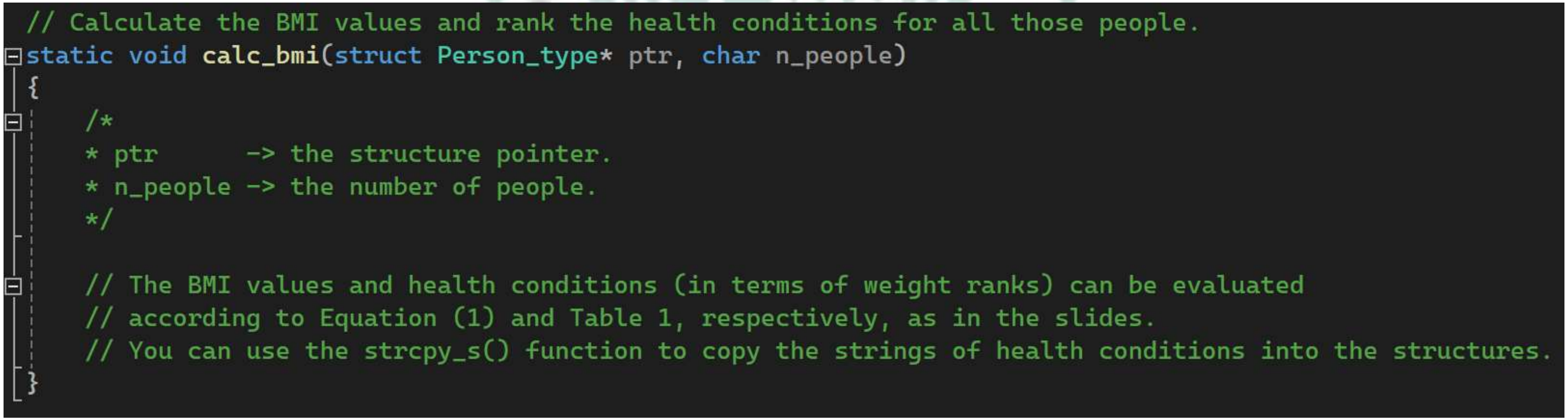
```c
// Structure definition
struct Person_type
{
    char name[10] = { 0 };
    char gender;
    float height;
    float weight;
    float bmi;
    float overweight;
    char health_condition[20] = { 0 };
};
```

- Advanced Part
- Program Structure
- 2, File reading



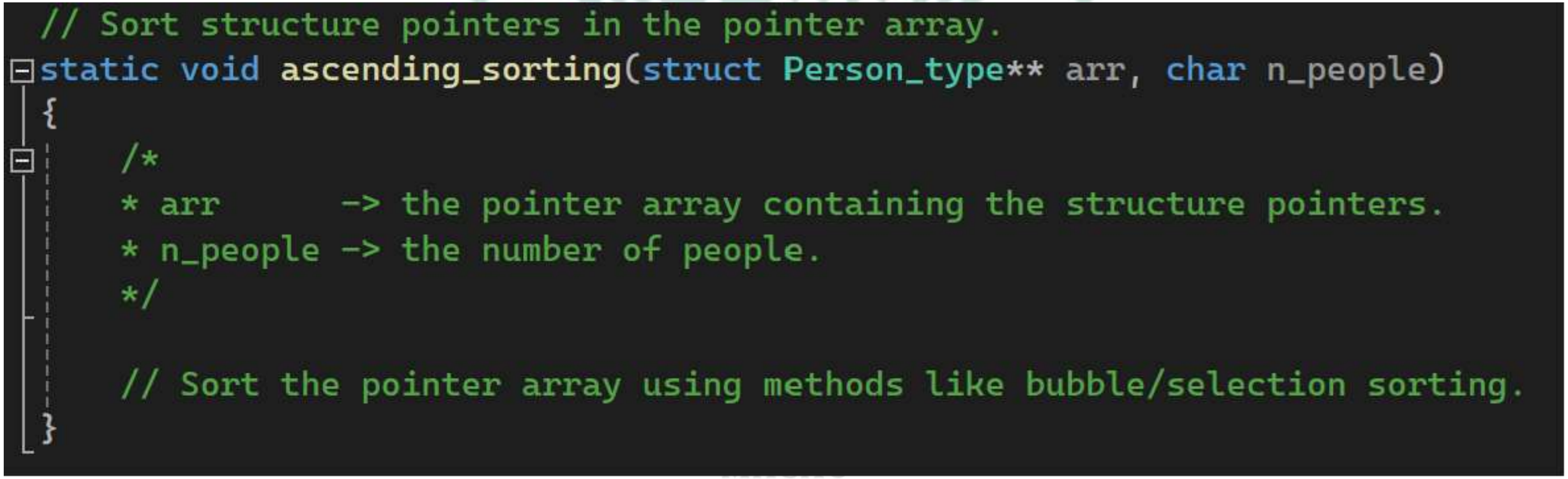
```
// File reading function
void read_file(struct Person_type* ptr, char n_people, char n_name)
{
    // open a file
    FILE* fp;
    errno_t err;
    if (err = fopen_s(&fp, "data.txt", "rb") != 0)
    {
        printf("Cannot open fhe file\n");
        exit(0);
    }
```

- Advanced Part
- Program Structure
- 3, Functions
- (1), BMI calculator function



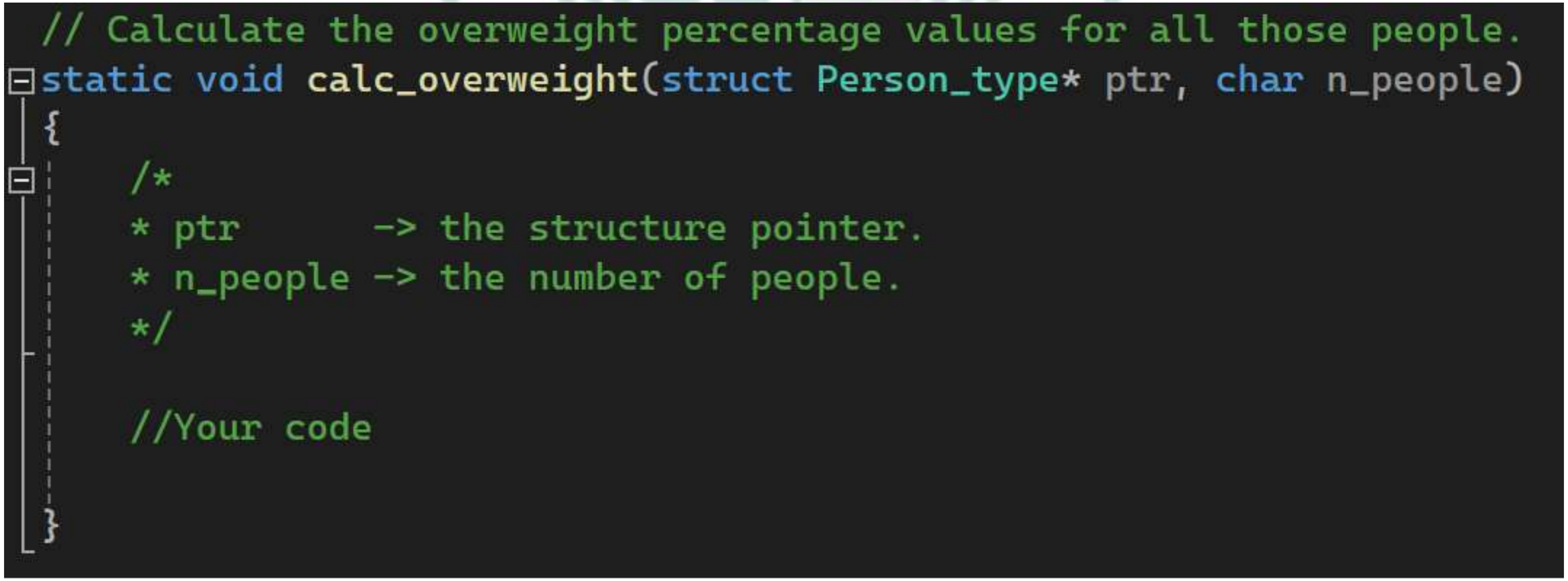
```c
// Calculate the BMI values and rank the health conditions for all those people.
static void calc_bmi(struct Person_type* ptr, char n_people)
{
    /*
    * ptr      -> the structure pointer.
    * n_people -> the number of people.
    */

    // The BMI values and health conditions (in terms of weight ranks) can be evaluated
    // according to Equation (1) and Table 1, respectively, as in the slides.
    // You can use the strcpy_s() function to copy the strings of health conditions into the structures.
}
```

- Advanced Part
- Program Structure
- 3, Functions
- (2), BMI sorting function



```c
// Sort structure pointers in the pointer array.
static void ascending_sorting(struct Person_type** arr, char n_people)
{
    /*
    * arr      -> the pointer array containing the structure pointers.
    * n_people -> the number of people.
    */

    // Sort the pointer array using methods like bubble/selection sorting.
}
```

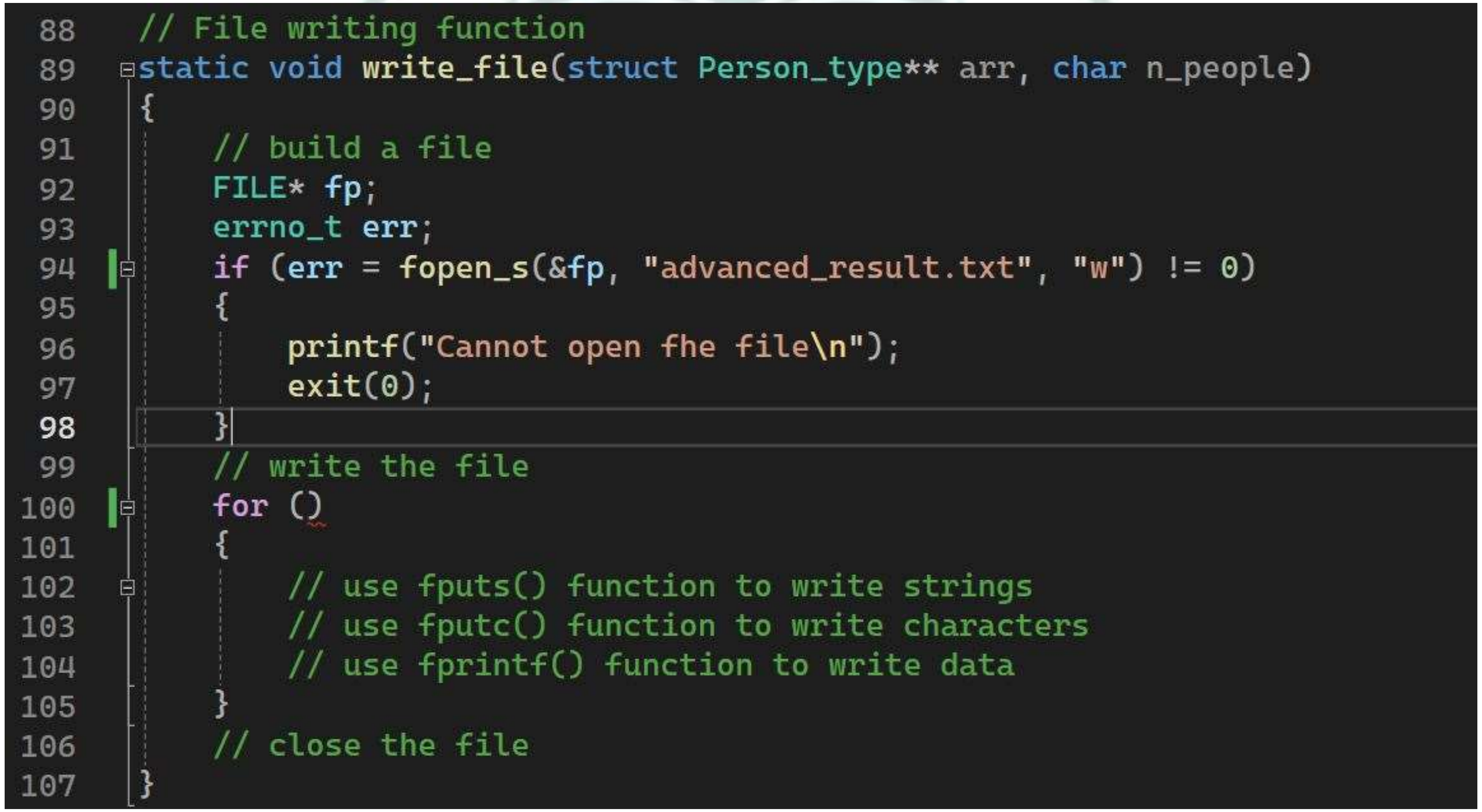
- Advanced Part
- Program Structure
- 3, Functions
- (3), overweight percentage calculator function



```
// Calculate the overweight percentage values for all those people.
static void calc_overweight(struct Person_type* ptr, char n_people)
{
    /*
    * ptr      -> the structure pointer.
    * n_people -> the number of people.
    */

    //Your code

}
```

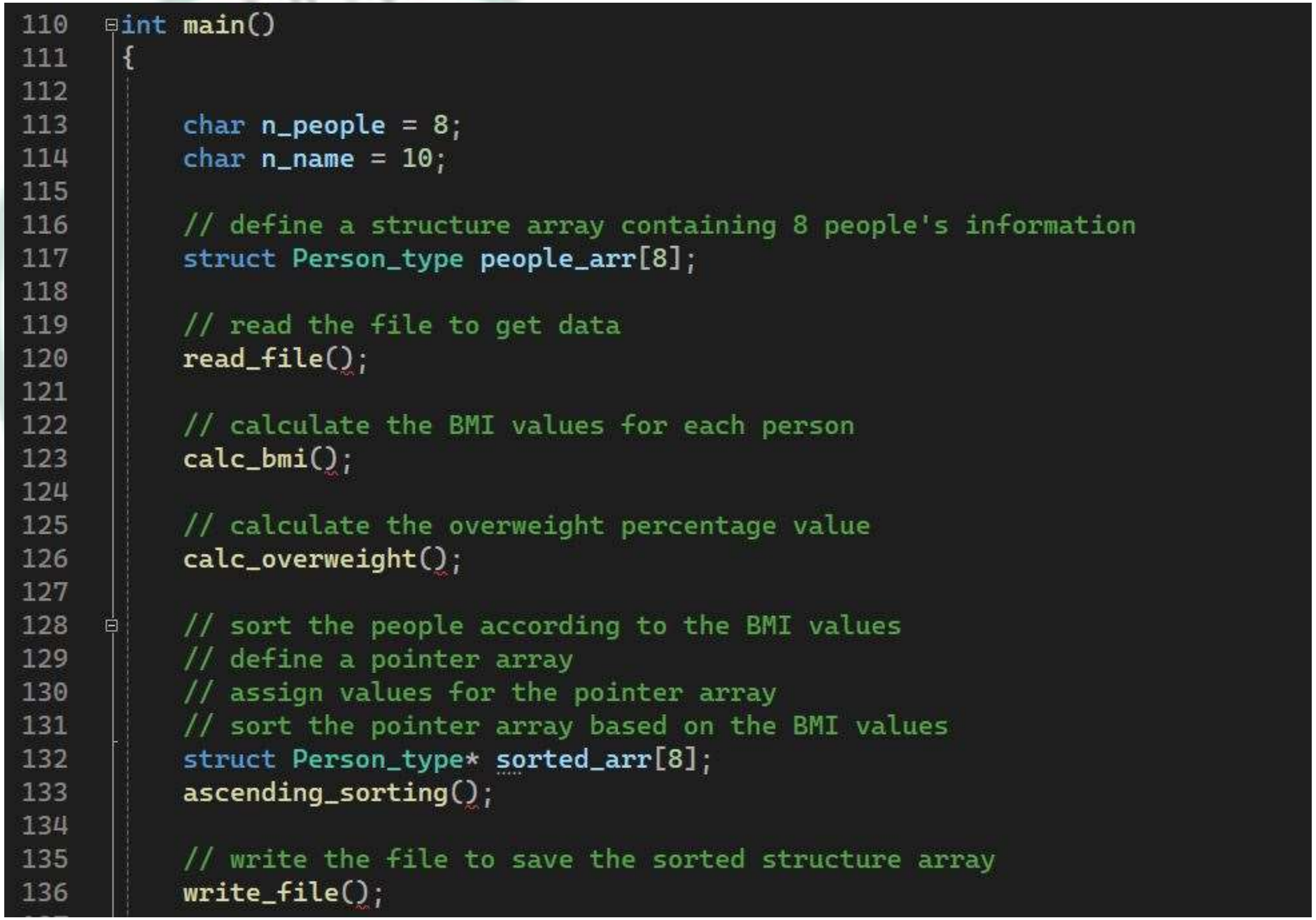
- Advanced Part
- Program Structure

- 4, File writing



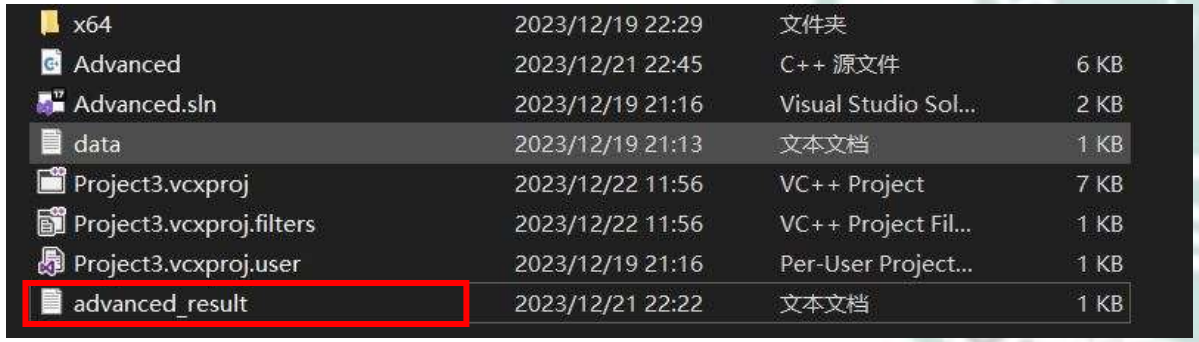
```
// File writing function
static void write_file(struct Person_type** arr, char n_people)
{
    // build a file
    FILE* fp;
    errno_t err;
    if (err = fopen_s(&fp, "advanced_result.txt", "w") != 0)
    {
        printf("Cannot open fhe file\n");
        exit(0);
    }
    // write the file
    for ()
    {
        // use fputs() function to write strings
        // use fputc() function to write characters
        // use fprintf() function to write data
    }
    // close the file
}
```

- Advanced Part
- Program Structure
- 5, main function



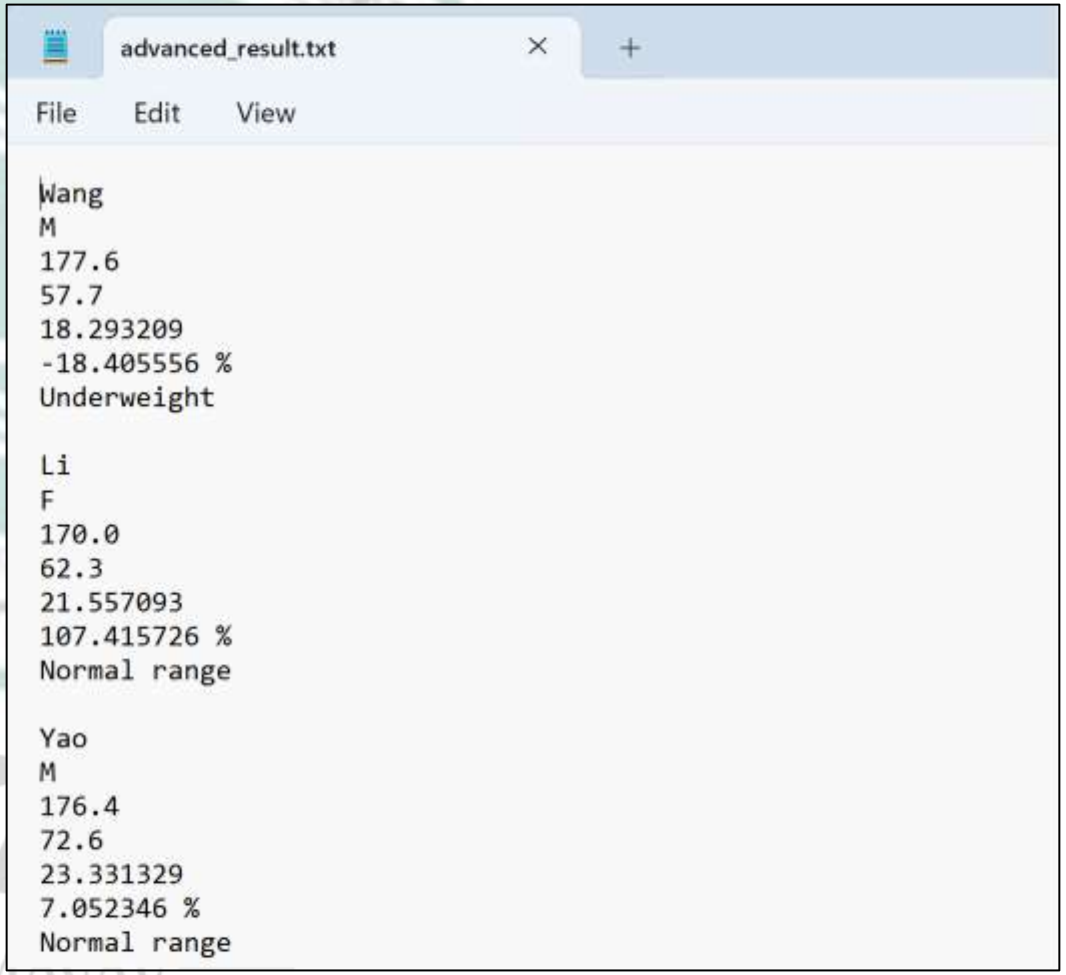
```
int main()
{

    char n_people = 8;
    char n_name = 10;

    // define a structure array containing 8 people's information
    struct Person_type people_arr[8];

    // read the file to get data
    read_file();

    // calculate the BMI values for each person
    calc_bmi();

    // calculate the overweight percentage value
    calc_overweight();

    // sort the people according to the BMI values
    // define a pointer array
    // assign values for the pointer array
    // sort the pointer array based on the BMI values
    struct Person_type* sorted_arr[8];
    ascending_sorting();

    // write the file to save the sorted structure array
    write_file();
```

- Advanced Part
- Outcomes after Program Execution



```
x64                        2023/12/19 22:29   文件夹
Advanced                   2023/12/21 22:45   C++ 源文件            6 KB
Advanced.sln               2023/12/19 21:16   Visual Studio Sol...  2 KB
data                       2023/12/19 21:13   文本文档              1 KB
Project3.vcxproj           2023/12/22 11:56   VC++ Project          7 KB
Project3.vcxproj.filters   2023/12/22 11:56   VC++ Project Fil...   1 KB
Project3.vcxproj.user      2023/12/19 21:16   Per-User Project...   1 KB
advanced_result            2023/12/21 22:22   文本文档              1 KB
```

The output file "advanced_result.txt"



```
Wang
M
177.6
57.7
18.293209
-18.405556 %
Underweight

Li
F
170.0
62.3
21.557093
107.415726 %
Normal range

Yao
M
176.4
72.6
23.331329
7.052346 %
Normal range
```

- Files for Submission

➢Basic part (40%)

- (70%) ***basic.c*** : a (single) C source file.

- (10%) ***basic_record.mp4*** : a short video recording the whole process of program execution, from starting running the program until the terminal window pops out displaying the result.

- (20%) ***basic_description.pdf*** : a PDF file describing your C source code, including the defined variables, arrays, functions, and the algorithms used, etc.

- Files for Submission

➢Advanced part (60%)

- (70%) ***advanced.c*** : a (single) C source file.
- (5%) ***advanced_result.txt*** : a result file writing the execution result.
- (10%) ***advanced_record.mp4*** : a short video file recording the whole process of program execution, beginning from where the project folder contains the *data.txt* but does **NOT** contain the *advanced_result.txt*, until the *advanced_result.txt* appears and shows the written contents after being double-clicked, then stop recording.
- (15%) ***advanced_description.pdf*** : a PDF file describing your C source code, including the defined variables, arrays, functions, and the algorithms used, etc.

- Notice

1. The names of all the seven files for submission should be set as instructed, and no file names should be used other than *basic.c*, *basic_record.mp4*, …, *advanced.c*, *advanced_result.txt*, … (otherwise, it may cause confusion and potential deduction of your marks).
2. All the seven files for submission from the two parts need to be submitted **separately (no folders needed) onto TronClass.**
3. This is an **individual project**, and please, **NO plagiarism (from others or ChatGPT-like tools)!**
4. The deadline is **23:55, 7th Jan, 2024**.