Advanced Description 代码解释

日期：

- 2024年1月3日

环境：

- 操作系统：Windows 11
- 集成开发环境：Microsoft Visual Studio 2022

源代码：

```
#include<stdio.h>
#include<errno.h>
#include<string.h>
#include<stdlib.h>
#include<math.h>

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


// 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.

    for (int i = 0; i < n_people; i++) {
        // Calculating BMI
        ptr[i].bmi = ptr[i].weight / pow((ptr[i].height / 100), 2);

        // Determining health condition based on BMI
        if (ptr[i].bmi < 18.5) strcpy_s(ptr[i].health_condition, "Underweight");
```

```
        else if (ptr[i].bmi < 24) strcpy_s(ptr[i].health_condition, "Normal
range");
        else if (ptr[i].bmi < 27) strcpy_s(ptr[i].health_condition,
"Overweight");
        else if (ptr[i].bmi < 30) strcpy_s(ptr[i].health_condition, "Mild
obesity");
        else if (ptr[i].bmi < 35) strcpy_s(ptr[i].health_condition, "Moderate
obesity");
        else strcpy_s(ptr[i].health_condition, "Severe obesity");

    }
}


// 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.
        */
        for (int i = 0; i < n_people; i++) {
            float standard_weight = (ptr[i].gender == 'M') ? 22 *
pow((ptr[i].height / 100), 2) : 21 * pow((ptr[i].height / 100), 2);
            ptr[i].overweight = ((ptr[i].weight - standard_weight) /
standard_weight) * 100;
        }
    }


// 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.
        for (int i = 0; i < n_people - 1; i++) {
            for (int j = 0; j < n_people - i - 1; j++) {
                if (arr[j]->bmi > arr[j + 1]->bmi) {
                    struct Person_type* temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
    }

// 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 data file\n");
        exit(0);
    }
    // read the file
    for (int i = 0; i < n_people; i++)
    {
        fgets(ptr[i].name, n_name, fp);
        ptr[i].name[strcspn(ptr[i].name, "\n")] = 0;
        // use fgets() function to read strings.
    }
    for (int i = 0; i < n_people; i++)
    {
        ptr[i].gender = fgetc(fp);
        fgetc(fp);
        // use fgetc() function to read characters.
        // use fgetc() function to get a "\n" to eliminate.
    }
    for (int i = 0; i < n_people; i++)
    {
        fread(&ptr[i].height, sizeof(float), 1, fp);
        fread(&ptr[i].weight, sizeof(float), 1, fp);
        // use fread() function to read data.
    }
    // close file
}


// 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 result file\n");
        exit(0);
    }
    // write the file
    for (int i = 0; i < n_people; i++)
    {
        fputs(arr[i]->name, fp);
        fprintf(fp, "\n");
        fputc(arr[i]->gender, fp);
        fprintf(fp, "\n");
        fprintf(fp, "%.1f\n%.1f\n%f\n%f%%\n%s\n\n", arr[i]->height, arr[i]-
>weight,
            arr[i]->bmi, arr[i]->overweight, arr[i]->health_condition);
        // use fputs() function to write strings
        // use fputc() function to write characters
        // use fprintf() function to write data
    }
    // close the file
};
```

```c
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(people_arr, n_people, n_name);

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

    // calculate the overweight percentage value
    calc_overweight(people_arr, n_people);

    // 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];
    for (int i = 0; i < n_people; i++) {
        sorted_arr[i] = &people_arr[i];
    }
    ascending_sorting(sorted_arr, n_people);

    // write the file to save the sorted structure array
    write_file(sorted_arr, n_people);

    return 0;
}
```

解释：

结构体定义

- `struct Person_type`：一个结构体，用于存储个人的健康信息，包括姓名、性别、身高、体重、BMI、超重百分比和健康状况。

函数

1. `calc_bmi`：
    - 功能：计算每个人的BMI值并确定其健康状况。
    - 参数：结构体指针`ptr`和人数`n_people`。
    - 算法：使用BMI公式计算BMI，并根据BMI的值判断健康状况。
2. `calc_overweight`：
    - 功能：计算每个人的超重百分比。
    - 参数：结构体指针`ptr`和人数`n_people`。
    - 算法：根据性别计算标准体重，然后计算超重百分比。

3. `ascending_sorting`：
    - 功能：根据BMI值对结构体指针数组进行升序排序。
    - 参数：结构体指针数组`arr`和人数`n_people`。
    - 算法：使用冒泡排序算法。
4. `read_file`：
    - 功能：从文件中读取数据并填充`people_arr`数组。
    - 参数：结构体指针`ptr`、人数`n_people`和姓名最大长度`n_name`。
5. `write_file`：
    - 功能：将排序后的数据写入文件。
    - 参数：结构体指针数组`arr`和人数`n_people`。

`main`函数

- 定义并初始化结构体数组`people_arr`。
- 从文件中读取数据填充`people_arr`。
- 计算每个人的BMI和超重百分比。
- 对人按BMI进行排序。
- 将排序后的数据写入新文件。