任务1补充

> 仅作参考，部分代码仍需自主完成。

各种功能都有很多实现方法，这里只说基础的方法，当然你用新标准或者其他的库也完全可以。

project_1

首先是生成随机数。在C++11之前，随机数通常由`cstdlib`库中的`rand()`函数生成，但注意这个随机数仅仅是一个伪随机数！它是根据一个数（我们可以称它为种子）为基准以某个递推公式推算出来的一系列数，当这系列数很大的时候，就符合正态公布，从而相当于产生了随机数，但这不是真正的随机数，当计算机正常开机后，这个种子的值是定了的，除非你破坏了系统。

如果需要变为广义的随机数，需要配合`srand(unsigned int seed)`函数设置 rand() 产生随机数时的随机数种子。参数 seed 必须是个整数，如果每次 seed 都设相同值，rand() 所产生的随机数值每次就会一样。可以使用`time()`传入不同的值。`time()`函数有返回值，即格林尼治时间`1970年1月1日00:00:00`到当前时刻的时长，时长单位是秒，用它作为种子。

```
#include <iostream>
#include <cstdlib>
#include <ctime>
```

(以下所有示例默认拥有上述头文件) 生成随机数：

```
int main() {
    srand(time(0));
    for (int i = 0; i != 5; ++i) {
        std::cout << rand() << std::endl;
    }
    return 0;
}
```

题目难度的控制可以从题目数量、数字范围、计算类型等方面入手。可以循环输出题目，通过改变循环变量的值控制题目数量。

```
int number = 10;  // 题目数量
int answer = 0;   // 回答值
std::cout << "[TIPS] 请依次完成以下 " << number << " 道加减法计算题: "
<< std::endl;
for (int i = 0; i != number; ++i) {
    // ......
    std::cin >> answer;
    // ......
}
```

数字范围可以通过限制随机数范围实现：

- `(rand() % (max-min))+ min;`取得 [min,max) 的随机整数
- `(rand() % (max-min+1))+ min;`取得 [min,max] 的随机整数
- `(rand() % (max-min))+ min+ 1;`取得 (min,max] 的随机整数

比如将随机数限制在[2,20]区间内：

```
srand(time(0));
int rvalue = 0;
int min = 2;
int max = 20;
for (int i = 0; i != 5; ++i) {
    rvalue = ( rand() % (max - min + 1) ) + min ;
    std::cout << rvalue << std::endl;
}
```

计算类型可以选择加减法、乘除法运算，只需创建相应的变量或数组存储运算结果（即正确答案）即可。以加法、乘法为例：

```
int add_answer = 0;
int add_result_array[number] = {};
int add_answer_array[number] = {};
int mul_answer = 0;
int mul_result_array[number] = {};
int mul_answer_array[number] = {};
for (int i = 0; i != number; ++i) {
    // ......
    add_result_array[i] = r1 + r2;      // r1、r2为生成的随机数
    mul_result_array[i] = r1 * r2;
    add_answer_array[i] = add_answer;
    mul_answer_array[i] = mul_answer;
}
```

答题后的反馈信息包括：答对或答错的题目数量、答错的题目与其正确答案。可以创建对应的计数变量保存答对或答错的题目数量，使用数组记录错题的题号与其正确答案，待全部题目完成后循环输出。

```
int right_count = 0;  // 保存正确结果数量
int wrong_count = 0;  // 保存错误结果数量
int result_array[number] = {};      // 保存标准答案
int answer_array[number] = {};      // 保存用户计算结果
int wrong_num_array[number] = {};  // 保存错题题号
```

project_2

关于计时，有很多方法，比如C/C++中的计时函数`clock()`。函数返回从“开启这个程序进程”到“程序中调用`clock()`函数”时之间的`CPU`时钟计时单元（clock tick）数。`CLOCKS_PER_SEC`表示一秒钟内CPU运行的时钟周期数（时钟计时单元）。

```
#include <ctime>
void fun() {};
int main() {
    clock_t start,end;  // 定义clock_t变量
    start = clock();    // 开始时间
    fun();              // 需计时的函数
    end = clock();      // 结束时间
    cout<<"time = "<<double(end-start)/CLOCKS_PER_SEC<<"s"<<endl;
 //输出时间（单位：s）
}
```

题目中需要输出每道题的计算时间，那就将这段代码加到每轮循环中（以加法为例）：

```
clock_t start,end;
int rvalue_1 = 0;
int rvalue_2 = 0;
int min = 2;
int max = 20;
int result_array[5] = {};
int answer_array[5] = {};
int answer = 0;
for (int i = 0; i != 5; ++i) {
    rvalue_1 = ( rand() % (max - min + 1) ) + min ;
    rvalue_2 = ( rand() % (max - min + 1) ) + min ;
    result_array[i] = rvalue_1 + rvalue_2;
    std::cout << rvalue_1 << " + " << rvalue_2 << " = " <<
std::endl;
```

```
    std::cout << "please enter the result: " << std::endl;
    start = clock();
    std::cin >> answer;
    answer_array[i] = answer;
    end = clock();
    std::cout<<"time = "<<double(end-start)/CLOCKS_PER_SEC<<"s"
<<std::endl;  //输出时间（单位：s）
}
```

此外还需要统计每道题的用时来计算最大用时和平均用时，可以用数组去实现。也就是说我们可以将每次的计算用时存入对应数组，最后输出平均值、最大值即可。

关于拓展部分，看完前面的代码你会发现它很繁琐，有很多重复的逻辑，修改起来也很麻烦。将它们封装成函数是一种好方法。比如封装函数用于生成随机数：

```
// 题目难度枚举
typedef enum {
    SIMPLE = 1,
    COMMON,
    HARD
}Difficulty;

// 功能：生成随机数
// 参数：题目难度
// 返回：随机数
int CreateRand(Difficulty difficulty) {
    switch (difficulty) {
        case SIMPLE:
            return rand() % (49 +1);  // [0,49]
        case COMMON:
            return ((rand() % (100 - 50)) + 50);  // [50,100)
        case HARD:
            return ((rand() % (100 - 0)) + 0);  // [0,100)
    }
}
```

当然如果你还是觉得现在的cpp文件头重脚轻，你可以将函数的声明和定义分别写入对应的头文件和源文件，这样会更标准、简便。