实验 6 排序算法

计算机类3班 施家鑫 25020007105

Content

C语言实现

> 随机生成 100000个随机数，进行冒泡排序和快速排序，并比较执行时间。

生成100000个随机数

- 引用头文件

```
#include <stdlib.h>
// 使用其中的rand()函数
#include <time.h>
// 记录执行时间
// 使用其中的time()函数为随机数做种子，以保证“随机”
```

- 使用time()为随机数做种

```
srand(time(NULL));
```

- 生成100000个随机数

```
int a[100001] = {0};
for (int i = 0; i < 100000; i++) {
    a[i] = rand() % 1000000;
}
```

实现冒泡排序功能

```
void bubble_sort(int a[], int n) {
    int i, j, temp;
    for (i = 0; i < n-1; i++) {
        for (j = 0; j < n-i-1; j++) {
            if (a[j] > a[j+1]) {
                temp = a[j];
                a[j] = a[j+1];
                a[j+1] = temp;
            }
        }
    }
}
```

实现快速排序功能

```
void quick_sort(int a[], int low, int high) {
    if (low < high) {
        int pivot = a[high];
        int i = low - 1;
        for (int j = low; j < high; j++) {
            if (a[j] < pivot) {
                i++;
                int temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
        }
        int temp = a[i + 1];
        a[i + 1] = a[high];
        a[high] = temp;
        int pi = i + 1;
        quick_sort(a, low, pi - 1);
        quick_sort(a, pi + 1, high);
    }
}
```

分别记录两者执行时间，并进行比较

- 使用两个变量分别记录开始与结束时间

```
clock_t start, end;
```

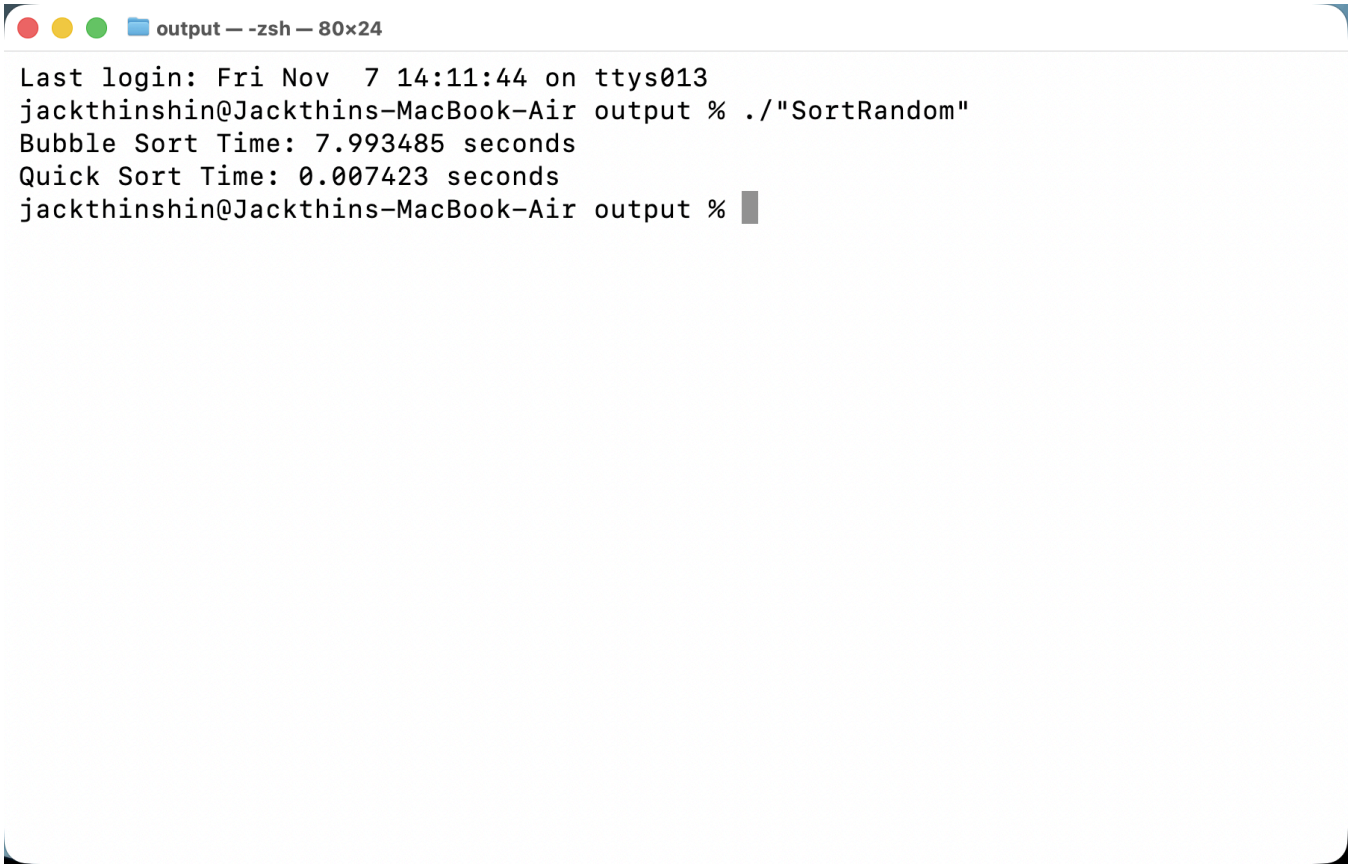
- 在排序功能执行的代码前后分别加上
  `start = clock();`
  `end = clock();`
  记录对应的时间
- 将时间转化为以秒为单位

```
double T = ((double)(end - start)) / CLOCKS_PER_SEC;
```

完成主程序

```
int main() {
    int a[100001] = {0};
    srand(time(NULL));
    for (int i = 0; i < 100000; i++) {
        a[i] = rand() % 1000000;
    }
    clock_t start, end;
    start = clock();
    bubble_sort(a, 100000);

    end = clock();
    double T = ((double)(end - start)) / CLOCKS_PER_SEC;
    printf("Bubble Sort Time: %f seconds\n", T);

    for (int i = 0; i < 100001; i++) {
        a[i] = rand() % 1000000;
    }

    start = clock();
    quick_sort(a, 0, 99999);
    end = clock();
    T = ((double)(end - start)) / CLOCKS_PER_SEC;
    printf("Quick Sort Time: %f seconds\n", T);

    return 0;
}
```

- 最终效果



Python语言实现

> 随机生成 100000个随机数，进行冒泡排序和快速排序，并比较执行时间。

思路(与C语言类似):

- 生成随机数功能
- 冒泡排序功能
- 快速排序功能
- 记录执行时间，并进行比较
- 主程序

生成100000个随机数到数组

导入time包

```
import time
```

按照 Python 语言规范放入数组

```
a = [random.randint(0, 100000) for _ in range(100000)]
```

冒泡排序功能

```
def bubble_sort(a: list):
    n = len(a)
    for i in range(n - 1):
        for j in range(0, n - 1 - i):
            if a[j] > a[j + 1]:
                a[j], a[j + 1] = a[j + 1], a[j]
```

快速排序功能

```
def quick_sort(a: list, low: int, high: int):
    if low < high:
        x = random.randint(low, high)
        a[x], a[high] = a[high], a[x]
        p = a[high]
        i = low - 1
        for j in range(low, high):
            if a[j] < p:
                i += 1
                a[i], a[j] = a[j], a[i]
        a[i + 1], a[high] = a[high], a[i + 1]
        p = i + 1
        quick_sort(a, low, p - 1)
        quick_sort(a, p + 1, high)
```

记录执行时间并进行比较

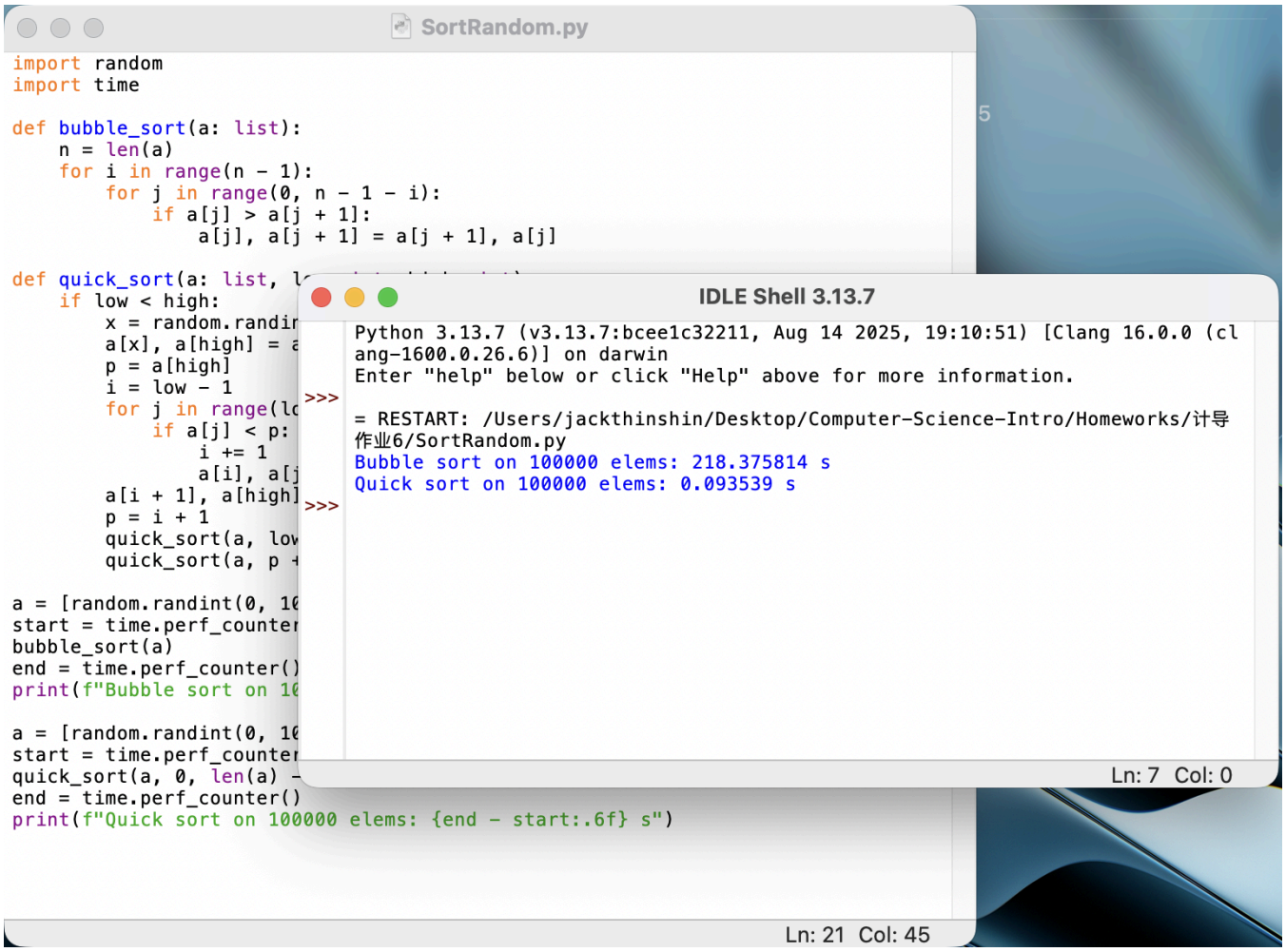
- 分别记录 start end 时间

```
start = time.perf_counter()
//排序操作
end = time.perf_counter()
```

主程序

```
a = [random.randint(0, 100000) for _ in range(100000)]
start = time.perf_counter()
bubble_sort(a)
end = time.perf_counter()
print(f"Bubble sort on 100000 elems: {end - start:.6f} s")

a = [random.randint(0, 100000) for _ in range(100000)]
start = time.perf_counter()
quick_sort(a, 0, len(a) - 1)
end = time.perf_counter()
print(f"Quick sort on 100000 elems: {end - start:.6f} s")
```

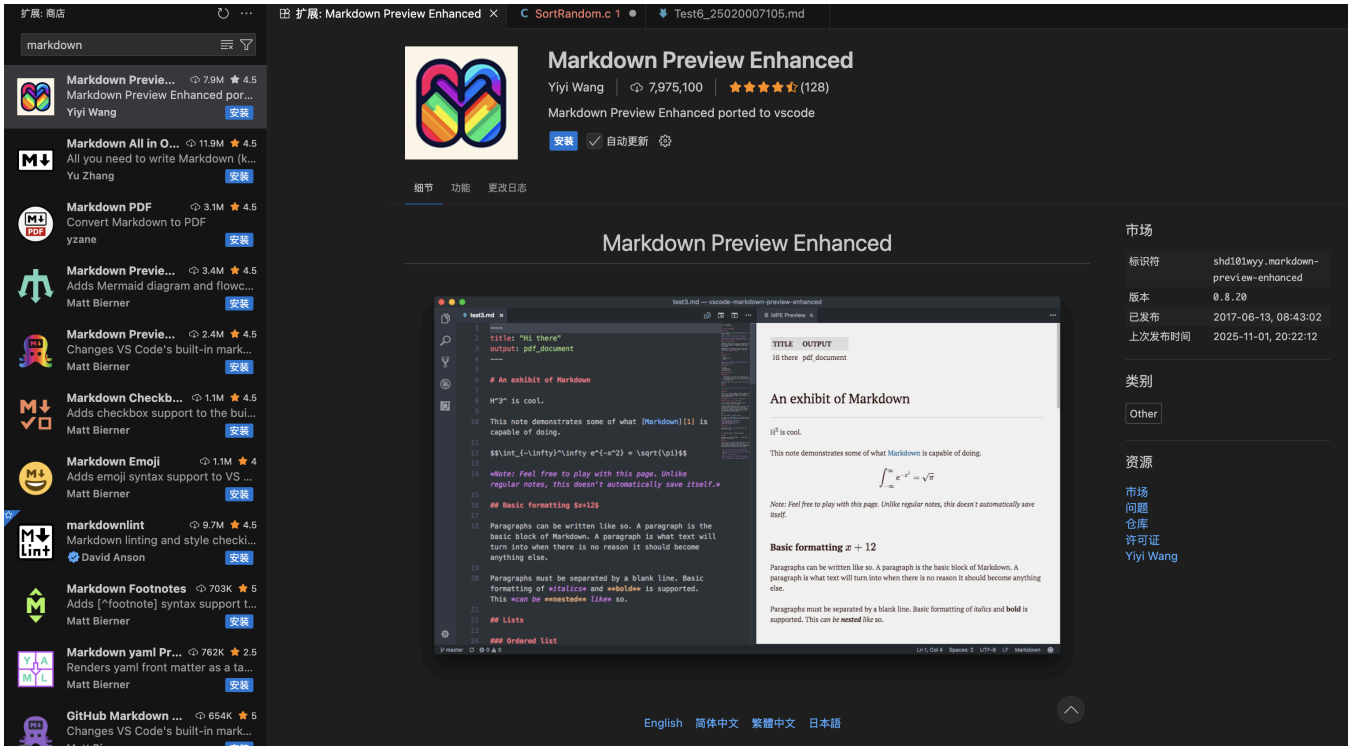
- 最终效果



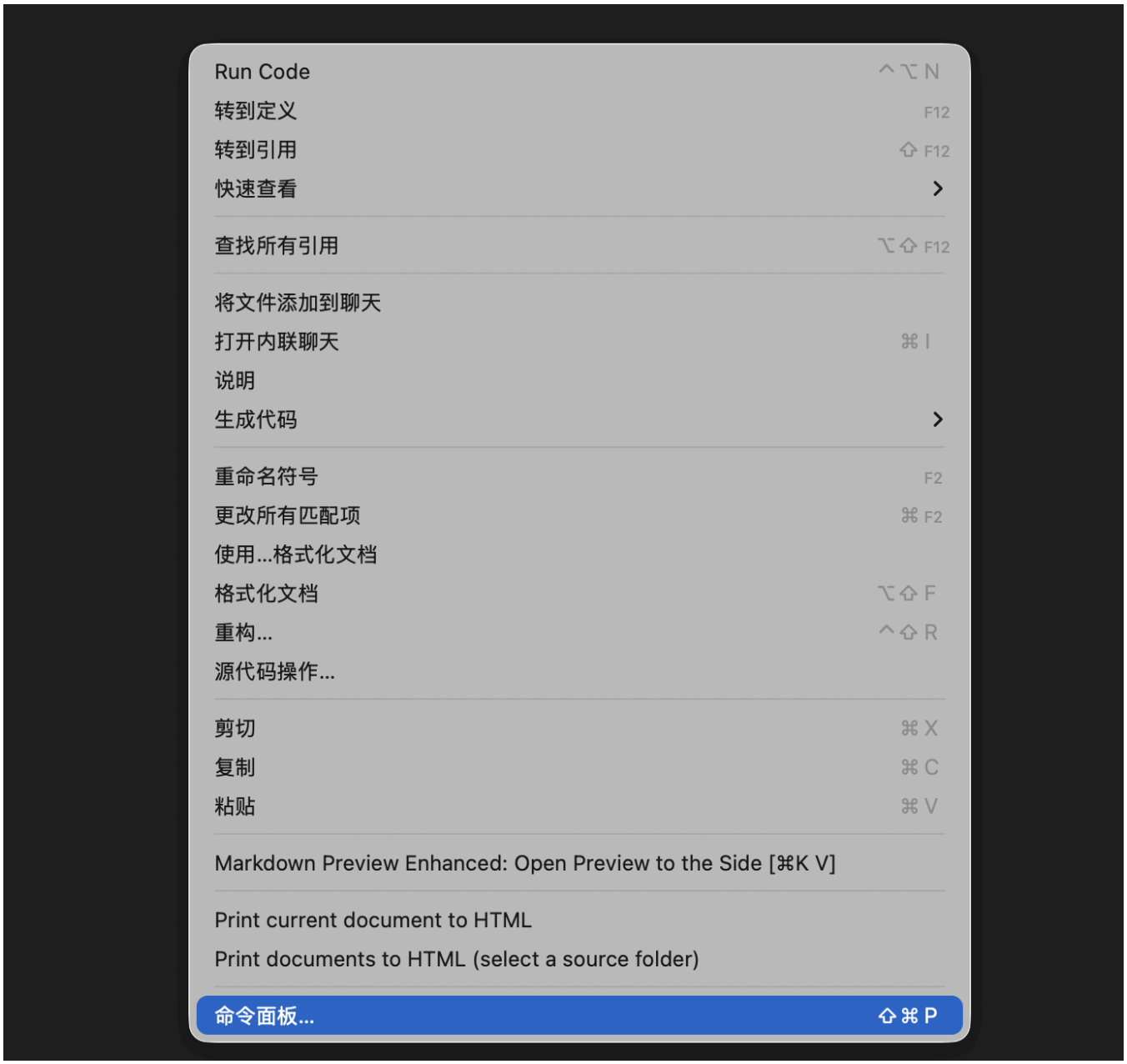
关于本文档的撰写

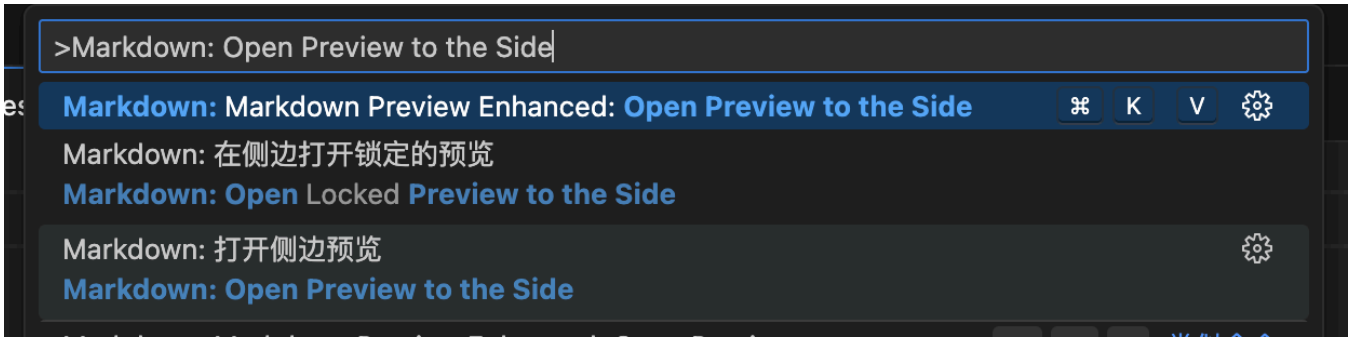
> 第一次使用Markdown

在Visual Studio Code中添加扩展



打开Side Preview以实现一边撰写一边预览效果的功能





最终工作区域实际效果如下：


计导作业6
SortRandom.c
SortRandom.py
Test6_25020007105.md 9+
Test6_25020007105.md > # 实验 6 排序算法 > ## Content
实验 6 排序算法
*** 计算机类3班 施家鑫 25020007105 ***
Content
Table of Contents (out of date)
- [C语言实现](#c语言实现)
- [生成100000个随机数](#生成100000个随机数)
- [实现**冒泡排序**功能](#实现冒泡排序功能)
- [实现**快速排序**功能](#实现快速排序功能)
- [分别记录两者**执行时间**，并进行**比较**](#分别记录两者执行时间并进行比较)
- [完成主程序](#完成主程序)
- [Python语言实现](#python语言实现)
- [生成100000个随机数到数组](#生成100000个随机数到数组)
- [冒泡排序功能](#冒泡排序功能)
- [快速排序功能](#快速排序功能)
- [记录执行时间并进行比较](#记录执行时间并进行比较)
- [主程序](#主程序)
- [关于本文档的撰写](#关于本文档的撰写)
C语言实现
> 随机生成 100000个随机数，进行冒泡排序和快速排序，并比较执行时间。
思路：
Table of Contents (out of date)
- [实现**生成随机数**功能](#生成100000个随机数)
- [实现**冒泡排序**功能](#实现冒泡排序功能)
- [实现**快速排序**功能](#实现快速排序功能)
- [分别记录两者**执行时间**，并进行**比较**](#分别记录两者执行时间并进行比较)
- [完成主程序](#完成主程序)
生成100000个随机数
- 引用头文件
预览 Test6_25020007105.md
实验 6 排序算法
计算机类3班 施家鑫 25020007105
Content
C语言实现
生成100000个随机数
实现冒泡排序功能
实现快速排序功能
分别记录两者执行时间，并进行比较
完成主程序
Python语言实现
生成100000个随机数到数组
冒泡排序功能
快速排序功能
记录执行时间并进行比较
主程序
关于本文档的撰写
C语言实现
随机生成 100000个随机数，进行冒泡排序和快速排序，并比较执行时间。
思路：
实现生成随机数功能
实现冒泡排序功能
实现快速排序功能
分别记录两者执行时间，并进行比较
完成主程序
生成100000个随机数
引用头文件
#include <stdlib.h>
// 使用其中的rand()函数
#include <time.h>
// 记录执行时间
// 使用其中的time()函数为随机数做种子，以保证“随机”
行 18，列 26 空格: 4 UTF-8 LF Markdown