实验 07：实现代办事项

一、实验目的

1、掌握 TypeScript 的基本概念；

2、掌握 TypeScript 的类型系统；

3、掌握使用 TypeScript 开发项目的方法。

二、实验学时

2 学时

三、实验类型

综合性

四、实验需求

1、硬件

每人配备计算机 1 台，建议优先使用个人计算机开展实验。

2、软件

安装 Visual Studio Code，以及 Edge 浏览器。

3、网络

本地主机能够访问互联网和实验中心网络。

4、工具

无。

五、实验任务

1、完成学生成绩管理系统示例；

2、完成代办事项实战。

六、实验内容及步骤

1、完成学生成绩管理系统示例

步骤 1：创建 grade-management 文件夹，并使用 Visual Studio Code 打开文件夹，如图 1 所示。

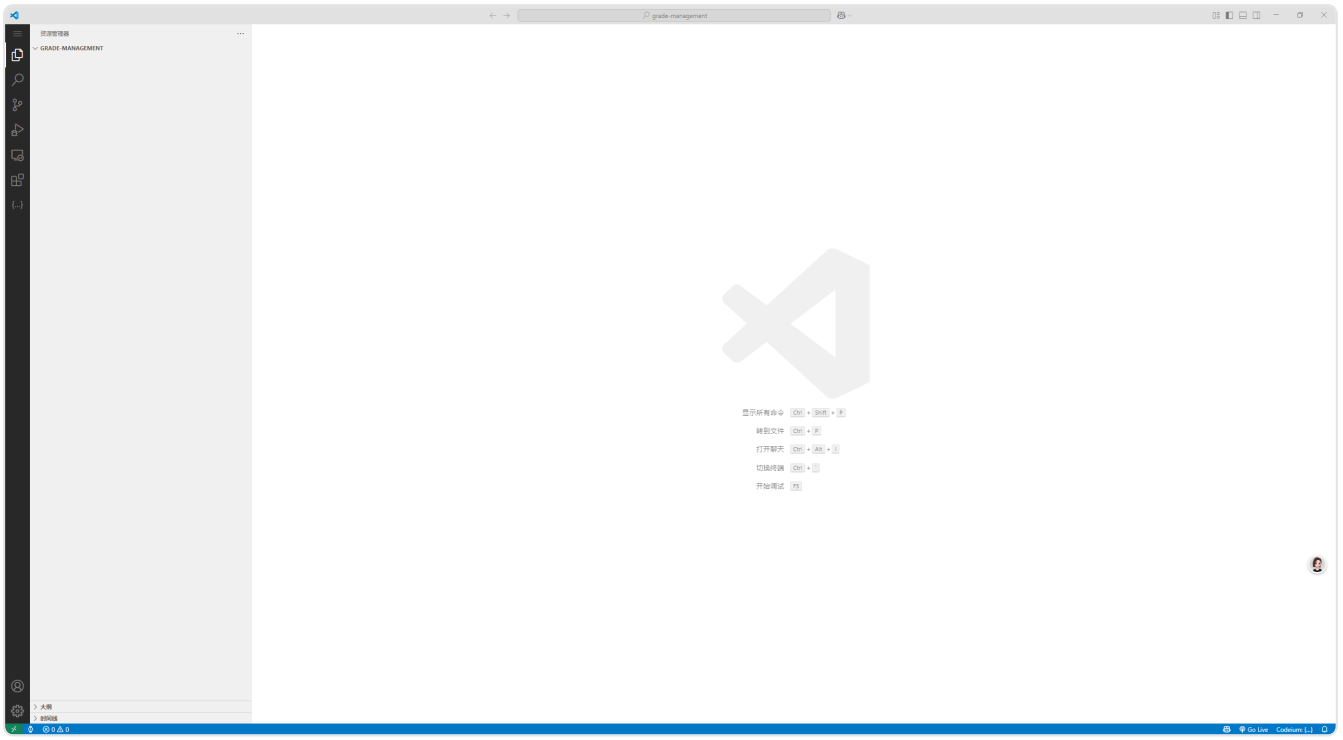

图1 使用Visual Stidio Code打开文件夹

步骤 2：在 Visual Studio Code 中打开终端，并执行以下命令进行项目初始化，如图 2 所示。

```Bash
1 npm init -y
```

```
PS C:\Users\admain\Desktop\grade-management> npm init -y
Wrote to C:\Users\admain\Desktop\grade-management\package.json:

{
  "name": "grade-management",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}


PS C:\Users\admain\Desktop\grade-management>
```

图2 执行项目初始化命令

步骤 3：执行以下命令安装项目依赖，执行结果如图 3 所示。

Bash

```
1 npm install commander
2 npm install typescript @types/node @types/commander --save-dev
```

```
PS C:\Users\admain\Desktop\grade-management> npm install commander

added 1 package, and audited 2 packages in 2s

found 0 vulnerabilities
PS C:\Users\admain\Desktop\grade-management> npm install typescript @types/node @types/commander --save-dev
npm warn deprecated @types/commander@2.12.5: This is a stub types definition. commander provides its own type definitions, so you do not need this installed.

added 4 packages, and audited 6 packages in 5s

found 0 vulnerabilities
PS C:\Users\admain\Desktop\grade-management>
```

图3 安装项目依赖

步骤 4：执行以下命令，初始化 TypeScript 配置文件：tsconfig.json，并参照以下配置文件，修订 tsconfig.json。

Bash

```
1 npx tsc --init
```

JSON

```json
{
    "compilerOptions": {
        "target": "ES6",
        "module": "commonjs",
        "outDir": "./dist",
        "rootDir": "./src",
        "strict": true,
        "esModuleInterop": true,
        "skipLibCheck": true,
        "forceConsistentCasingInFileNames": true
    },
    "include": ["src/**/*.ts"]
}
```

步骤 5：在项目根目录下创建 src 目录，用于存放 TypeScript 源代码。

步骤 6：在 src 目录下新建 studentGradeManagement.ts 文件，实现学生成绩管理，并参照以下代码撰写。

TypeScript

```typescript
import * as fs from 'fs';
import * as path from 'path';

// 定义学生成绩接口
interface StudentGrade {
    id: number;
    name: string;
    subject: string;
    grade: number;
}

// 定义学生成绩管理类
class StudentGradeManagement {
    private grades: StudentGrade[] = [];
    private dataFilePath = path.join(__dirname, 'grades.json');

    constructor() {
        this.loadGradesFromFile();
    }

    // 从文件中加载学生成绩数据
    private loadGradesFromFile() {
        try {
            if (fs.existsSync(this.dataFilePath)) {
                const data = fs.readFileSync(this.dataFilePath, 'utf
8');
                this.grades = JSON.parse(data);
            }
        } catch (error) {
            console.error('读取学生成绩数据文件时出错:', error);
        }
    }

    // 将学生成绩数据保存到文件
    private saveGradesToFile() {
        try {
            const data = JSON.stringify(this.grades, null, 2);
            fs.writeFileSync(this.dataFilePath, data, 'utf8');
        } catch (error) {
            console.error('保存学生成绩数据文件时出错:', error);
        }
    }
```

```
42
43      // 添加学生成绩
44      addGrade(name: string, subject: string, grade: number): StudentGrad
    e {
45          const newGrade: StudentGrade = {
46              id: this.grades.length > 0 ? this.grades[this.grades.lengt
    h - 1].id + 1 : 1,
47              name,
48              subject,
49              grade
50          };
51          this.grades.push(newGrade);
52          this.saveGradesToFile();
53          return newGrade;
54      }
55
56      // 删除学生成绩
57      removeGrade(id: number): boolean {
58          const index = this.grades.findIndex(grade => grade.id === id);
59          if (index !== -1) {
60              this.grades.splice(index, 1);
61              this.saveGradesToFile();
62              return true;
63          }
64          return false;
65      }
66
67      // 查询学生成绩
68      findGradeById(id: number): StudentGrade | undefined {
69          return this.grades.find(grade => grade.id === id);
70      }
71
72      // 显示所有学生成绩
73      getAllGrades(): StudentGrade[] {
74          return this.grades;
75      }
76  }
77
78  export { StudentGrade, StudentGradeManagement };
```

步骤 7：在 src 目录下编写 main.ts 文件，实现命令行操作，参考以下代码撰写。

TypeScript

```typescript
import { program } from 'commander';
import { StudentGrade, StudentGradeManagement } from './studentGradeManagement';

// 创建学生成绩管理实例
const gradeManager = new StudentGradeManagement();

// 配置命令行选项
program
    .version('1.0.0')
    .description('学生成绩管理系统命令行工具');

// 添加学生成绩命令
program
    .command('add <name> <subject> <grade>')
    .description('添加学生成绩')
    .action((name, subject, grade) => {
        const newGrade = gradeManager.addGrade(name, subject, Number(grade));
        console.log('成功添加学生成绩: ', newGrade);
    });

// 删除学生成绩命令
program
    .command('remove <id>')
    .description('根据 ID 删除学生成绩')
    .action((id) => {
        const isRemoved = gradeManager.removeGrade(Number(id));
        if (isRemoved) {
            console.log('成功删除学生成绩，ID 为: ', id);
        } else {
            console.log('未找到 ID 为', id, '的学生成绩');
        }
    });

// 查询学生成绩命令
program
    .command('find <id>')
    .description('根据 ID 查询学生成绩')
    .action((id) => {
        const foundGrade = gradeManager.findGradeById(Number(id));
        if (foundGrade) {
```

```
            console.log('查询到的学生成绩: ', foundGrade);
        } else {
            console.log('未找到 ID 为', id, '的学生成绩');
        }
    });

// 显示所有学生成绩命令
program
    .command('list')
    .description('显示所有学生成绩')
    .action(() => {
        const allGrades = gradeManager.getAllGrades();
        console.log('所有学生成绩: ', allGrades);
    });

// 解析命令行参数
program.parse(process.argv);
```

步骤 8：执行以下命令，编译并运行项目。

Bash

```
npx tsc
```

步骤 9：运行编译后的代码，测试功能。

Bash

```
# 添加图书
node dist/main.js add "深入理解计算机系统" "Randal E. Bryant"
# 删除图书
node dist/main.js remove 1
# 查询图书
node dist/main.js find 2
# 显示所有图书
node dist/main.js list
```

2、实现代办事项实战

创建一个简单的待办事项命令行管理系统，支持添加、删除待办事项以及查看所有待办事项的功能。要求使用 TypeScript 编写，并且将待办事项数据存储在 JSON 文件中，其详细要求如下：

- 待办事项数据结构：定义一个 Todo 接口，包含 id（唯一标识）、title（待办事项标题）、completed（是否完成，布尔类型）三个属性。
- 待办事项管理类：创建一个 TodoManager 类，包含以下方法：
  - addTodo(title: string)：添加一个新的待办事项，自动生成唯一的 id，默认 completed 为 false。
  - removeTodo(id: number)：根据 id 删除对应的待办事项。
  - getAllTodos()：返回所有待办事项的数组。
  - markTodoAsCompleted(id: number)：将指定 id 的待办事项标记为已完成。
  - getCompletedTodos()：返回所有已完成的待办事项。
  - getIncompleteTodos()：返回所有未完成的待办事项。
- 文件操作：在 TodoManager 类的构造函数中，从 todos.json 文件中加载待办事项数据；在添加或删除待办事项后，将更新后的数据保存到 todos.json 文件中。
- 错误处理：在 TodoManager 类的各个方法中添加错误处理，例如当删除或标记不存在的待办事项时，输出错误信息。
- 输入验证：在命令行交互中，对用户输入的 id 进行验证，确保其为有效的数字；对添加的待办事项标题进行验证，确保不为空。
- 命令行交互：使用 commander 库实现命令行交互，支持以下命令：
  - add <title>：添加一个新的待办事项。
  - remove <id>：根据 id 删除待办事项。
  - list：显示所有待办事项。
  - complete <id>：将指定 id 的待办事项标记为已完成。
  - completed：显示所有已完成的待办事项。
  - incomplete：显示所有未完成的待办事项。

七、实验考核

本实验考核采用【实验随堂查】方式开展。

每个实验完成后，在实验课上通过现场演示的方式向实验指导教师进行汇报，并完成现场问答交流。

每个实验考核满分 100 分，其中实验成果汇报 60 分，现场提问交流 40 分。

实验考核流程：

（1）学生演示汇报实验内容的完成情况，实验指导老师现场打分。

（2）指导老师结合实验内容进行提问，每位学生提问 2-3 个问题，根据回答的情况现场打分。

（3）实验考核结束后，进行公布成绩。