程序设计学习的一些Tips

1 学习目的

计算思维

- 一些具体的编程技能
- 一些典型问题的求解

2 计算模型

硬件模型：CPU+RAM

- 数据和指令均存储在内存中
- 按照指令对数据进行操作

计算模型：信息（数据）+ 数据处理

- 如何表示信息？如何对信息进行建模？
  - 数值
  - 非数值类型
- 如何处理数据
  - 计算
  - 和设备交互：从外设获取数据，输出到设备（屏幕，发出声音等）
- 数据表示和处理往往是一起考虑的，如
  - 图像，声音，文字的编码和解码

3 数据的视角

如何对信息进行建模（数据表达）

- 内存管理角度
  - 静态，数据的数量不变
  - 动态，数据的数量发生变化
- 数据模型的复杂程度
  - 基础内置

    - 自定义和组合

3.1 静态与动态内存

3.1.1 静态内存

使用内置类型，或者自定义类型定义变量，或者数组

3.1.2 动态内存

指针（存放内存地址的变量，间接方式访问内存）

- 支持动态内存（长度不固定，运行时请求）（所申请内存的访问接口：使用和释放）
- 实现物理存储（内存中的表达）的非线性机制
    - 将群体实现成物理存储上（内存）的非连续结构，如实现非线性表达，以及动态的插入，删除数据
    - 数组的物理存储为连续方式

注意

- 指针一定要赋值【指向变量或者内存】，才能使用

3.2 基础类型与组合自定义类型

3.2.1 基础（内置）类型

变量（内存的标识符，直接访问）

- 变量名作为标识符，指代对应的内存。数据存储在内存中，能通过变量名访问。
- 数据抽象成数据类型，以消除具体问题的差异，程序具有一般性。
- 基本定义方式：通过内置的数据类型定义变量。需要保存多少信息，就需要定义相应多的变量。

基础的数据类型：单一类型

```
int count;
double pi;
char ch;
```

3.2.2 复合类型

表达群体：数组

表达内存地址：指针

```
int scores[23];
int *p = scores;
```

3.2.3 自定义类型

数据表达能力升级

- 表达复杂数据类型：结构体

```
struct StudentInfo
{
    char name[32];
    int age;
};
struct StudentInfo stu1;
```

3.2.4 自定义复合类型

```
struct StudentInfo
{
    char name[32];
    int age;
};
struct StudentInfo students[23];

struct StudentInfo *pInfo = &students[0];
```

3.3 变量的基础用法

- 初始化【申明的同时赋初值】
- 外部输入
  - 键盘：scanf(), fgets
  - 内存缓冲区：sscanf()
- 赋值

- 常数，变量，表达式
- 函数返回值

4 数据处理的视角

4.1 基础语句类型

三种类型的语句：顺序，分支，循环

- 顺序
  - 变量先申明，再使用
  - 变量先赋值，再使用

```
int weight = 10; //初始化
weight++;
```

- 分支

```
int flag;
//....
if(flag){
}
else{
}

char ch;
getchar(ch);
switch(ch){
case '1':
    break;
default:
    break;
}
```

- 循环

```
while(express){
}

for(express1; express2; express3){
}

do{
}while(express)
```

4.2 数据流

4.2.1 数据的输入和输出

- 输入到变量
  - 键盘输入
  - 从内存缓冲输入
  - 文件输入
- 变量输出
  - 输出到屏幕
  - 输出到内存缓冲
  - 输出到文件

4.2.2 数据数值的改变

- 赋值=
  - 常量，变量，表达式，函数返回值
- 指针运算*
  - 直接操作，函数参数方式

4.3 函数

- 函数三要素
  - 函数名，输入参数，输出参数
- 标记参数类型
  - 输入参数
    - 基础内置类型
    - 自定义类型：const *

        - 数组（指针）： const *
    - 输出参数
        - 数组（指针）: *
- 返回数据的方式
    - return【1个返回值，优先使用】
    - 指针【多于1个】
    - return+指针【return返回函数执行成功与否，指针返回计算结果】

5 为引入软件工程做准备

5.1 软件开发类型

两种类型的软件开发

- 基于项目。客户给出需求。
- 软件产品。

此外还有一些为了特定需求的类似于压缩软件的“小”软件。

5.2软件工程的步骤

通常，软件开发可以分为以下几个步骤：

1. 需求分析：确定软件的目标和功能，了解用户需求和期望，以及确定软件需要满足的要求和限制。
2. 设计阶段：基于需求分析的结果，设计软件的架构，包括数据结构、算法、接口、模块等，以及选择适合的技术栈和工具。
3. 编码阶段：根据设计文档编写代码，实现软件的各个模块和功能。
4. 测试阶段：通过单元测试、集成测试和系统测试等手段验证软件的正确性、可靠性和性能，同时检查是否满足用户需求和规范。
5. 部署和维护：将软件部署到生产环境，并确保软件的稳定运行，同时进行必要的维护和更新，以确保软件一直能够满足用户的需求。

除此之外，还有一些辅助性的工作，如文档编写、代码版本控制、团队协作等，这些都是软件开发过程中不可或缺的步骤。

5.3 更好的软件代码

一些观念需要建立

- 更“好”的程序
  - 可读性
  - 鲁棒性
  - 扩展性
  - 简洁【形式和逻辑上均简洁】
  - 通用性【代码可复用】
- 模块化
  - 内聚【模块内的不可分离性】
  - 耦合【模块间的依赖性】
  - 层次
- 抽象
  - 概括/简化，提取本质而忽略细节
  - 一般化/通用处理，升级
  - 控制抽象->函数
  - 数据抽象->向量，列表等
- 接口、信息隐藏
  - 【设计层面及宏观上逻辑上的视角】
  - 以接口方式定义模块功能以及交互（调用）方式
  - 对细节实现隐藏【可以是过程细节，也可以是结构细节，以及结构和过程的细节】
  - 可以是抽象方式实现

6. 一些有益的观念

6.1 软件模型可以采用三个视角

软件包含的数据对象：数据对象，属性，以及数据对象之间的关系。【考察实体，以及实体之间的关系】

软件的信息流动：信息留和数据从输入移动到输出的过程中所经受的变换。【数据在模块的流动】

软件的功能：系统分为若干个状态，以及对事件的相应。【操作相应，功能流程】

6.2 理清一些概念

1 数据对象与属性【数据建模角度】

数据对象是复合信息的表示。所谓复合信息是指具有一系列不同性质或属性的事物。

数据对象可以是外部实体（如产生或使用信息的任何事物）、事物（如报表或屏幕显示）、行为（如打电话）或事件（如响警报）、角色（如销售员）、单位（如会计科）以及地点（如仓库）或结构（如文件）等。

属性定义了数据对象的性质。

例如，为了开发机动车管理系统，描述汽车的属性应该是制造商、品牌、型号、发动机号码、车体类型、颜色、车主姓名、住址、驾驶证号码、生产日期、购买日期等。

2 行为【数据处理角度】

对数据对象或者属性进行操作

其中，数据对象彼此之间相互连接的方式称为关系，也称为联系。

6.3 程序设计的三重境界

- 看山是山
- 看山不是山。抽象
- 还是山。抽象和现象合一

6.4 学习方法

6.4.1 基础学习方法

- 不要“刷书”，按顺序学，缺少思维
- 应：
    - 带着问题去寻找解决方法和答案
    - 逐渐建立体系，理解深入
- 选择

    - 体系，启发
    - 而不是没有整理的知识点

6.4.2 怎么学

- 下真正的决心
    - 首先，内心里不能否定学会的可能性。（否则没有机会学好）
    - 逐渐发展成由内而外的喜欢和自信。
- 多看代码
    - 大脑逐渐熟悉。消除内心的陌生感和排斥感
- 多写代码
    - 熟能生巧，形成认知/技能记忆（类似肌肉记忆）
    - 不怕犯错，从错误中学习。教训是最宝贵的学习财富。
    - 不要复制-粘贴。需要锻炼手和大脑，让大脑学会如何读、写、看代码。如果只是复制-粘贴，只能骗自己，练习就白做了。
- 由内而外学习
    - 从问题出发，凑资源（找资料和学习），积少成多，善于使用80-20法则（20%精力解决80%问题）
    - 先实现，再完善，永远不够完美。（各个阶段代码都有相应价值）
    - 参考代码，多问问题。多刨根问底。多看官方文档。
- **Study Hard in a Smart Way**！
    - Hard Way（“笨办法”）【培养专业核心技能的工夫是不可能少的】
        - 不断**编码和调程序**。不断用程序解决问题
        - 遇上很多问题，迅速积累自己的**一手经验**
    - Smart Way
        - 多看、多思考、多借鉴，定期“复盘”
        - 多想。通过思考，关联知识和方法，总结得失，进一步提炼和抽象，扩展知识。需要停下来
        - 培养**良好的编程思维习惯**
- 拿着锤子找钉子？拿着钉子找锤子
    - 拿着锤子找钉子：单项练习，已知识点为线索的学习
    - 拿着钉子找锤子：根据要解决的问题凑素材，以问题为线索的学习