Chapter2

接口，API？大模型！

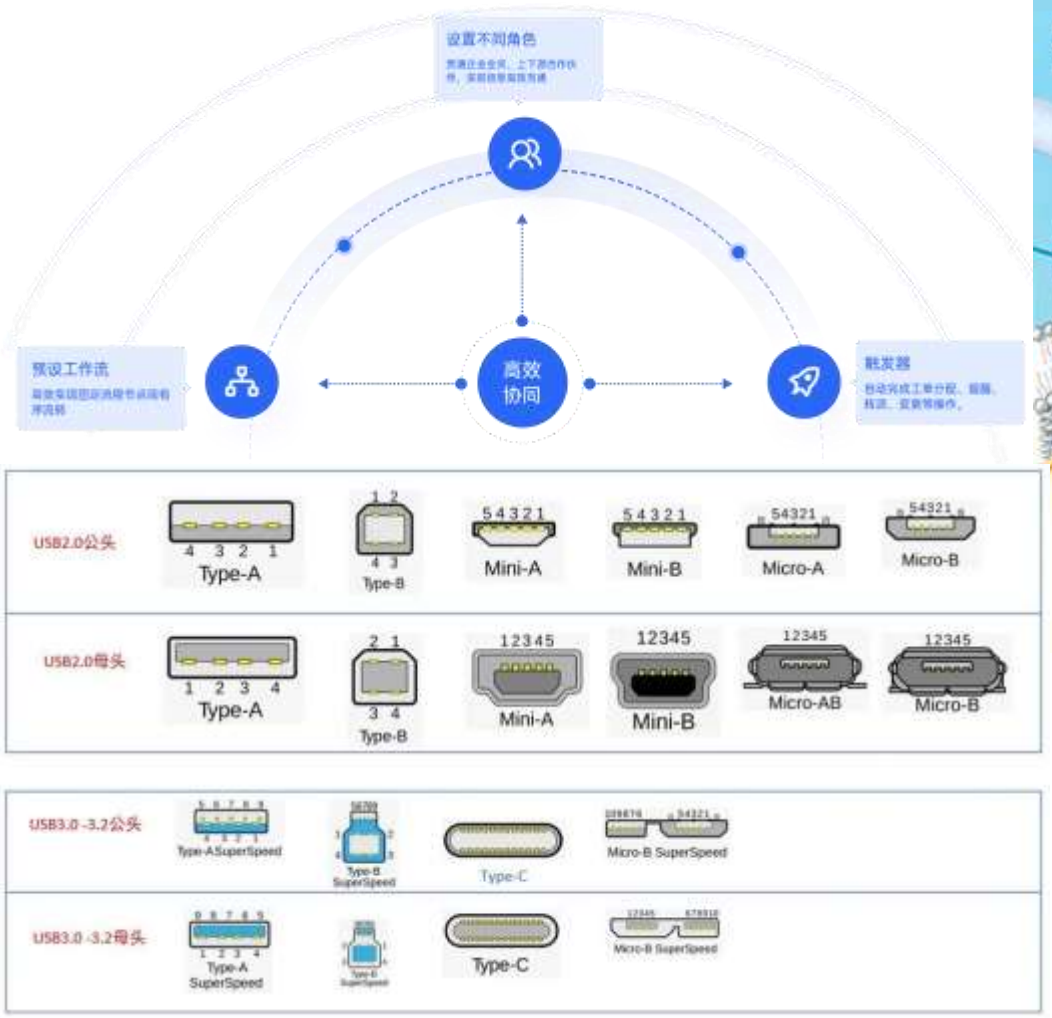
内容目录:

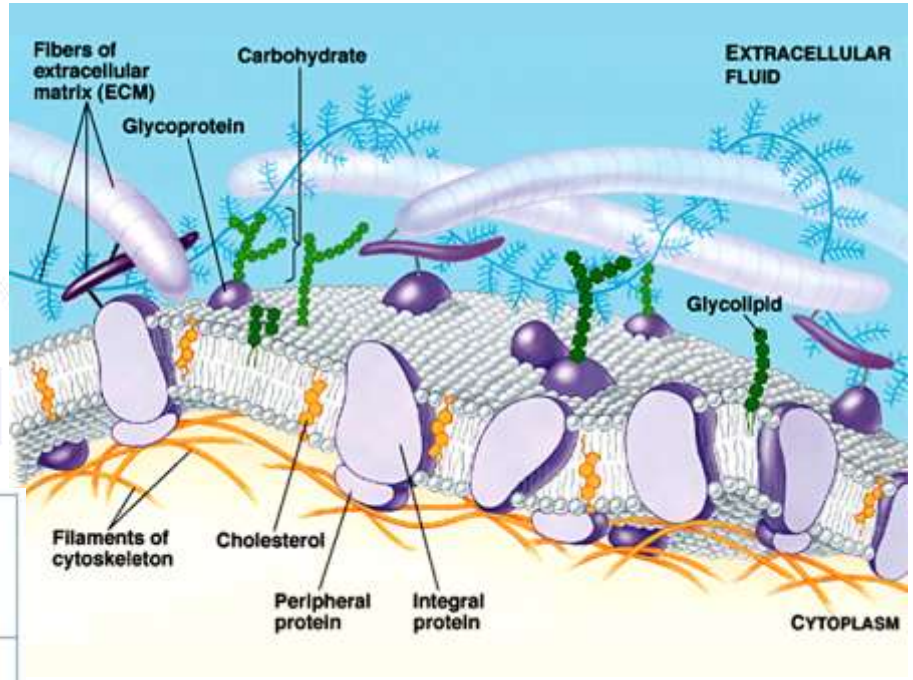
Think:

- “接口”这个词在哪里出现过?
  USB接口……

- 食堂吃饭的时候，是我们自己去做饭、打饭、缴费，还是通过专门的窗口、工作人员进行?

广义的接口

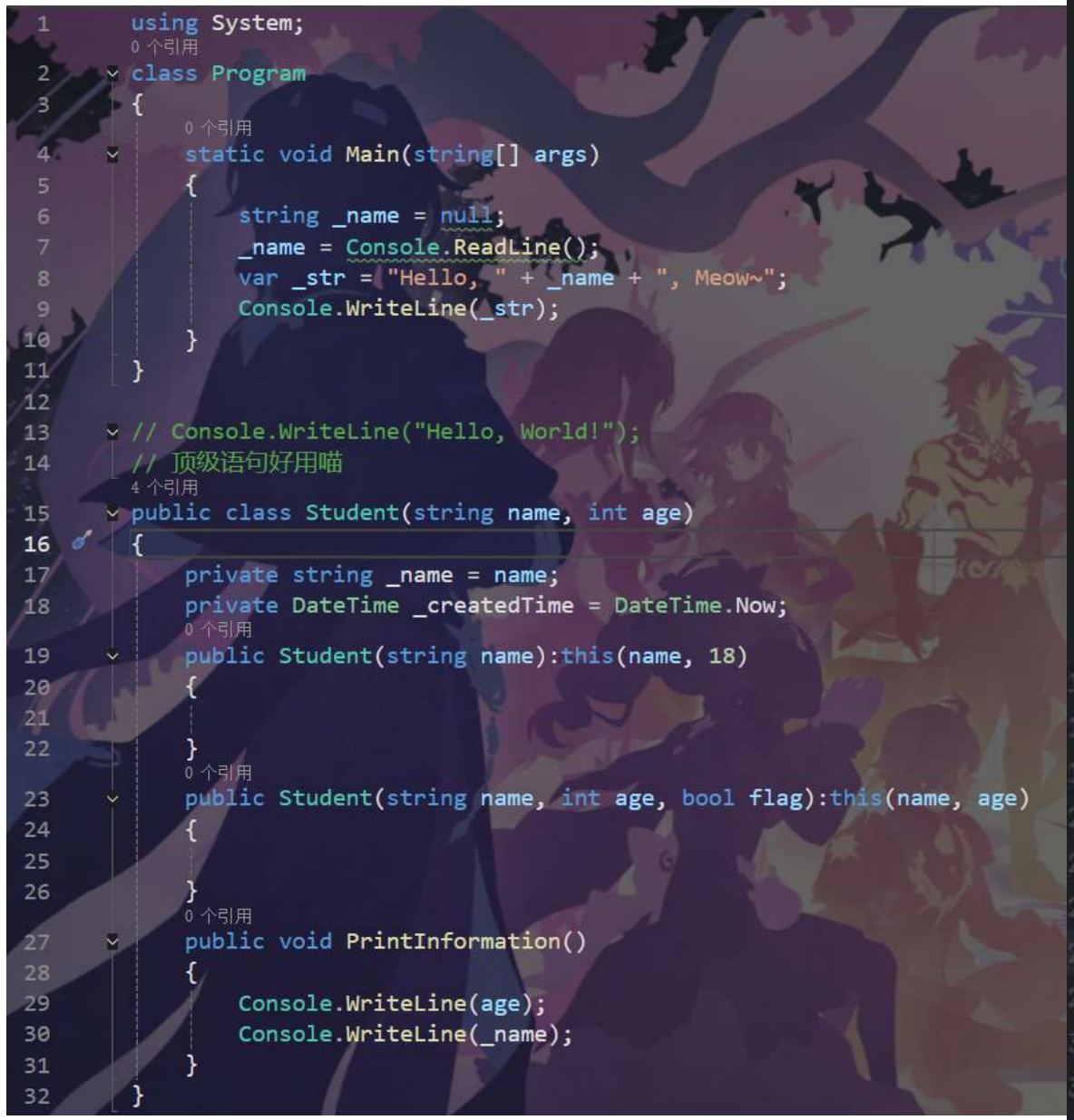




函数式编程（FP）、面向对象（OOP）
操作系统（OS）、用户界面（UI）

```cpp
#include <algorithm>
#include <sstream>
#include <cmath>
#include <climits>
#include <map>
#include <stack>
#include <queue>
#include <iomanip>
using namespace std;
struct plain {
    int width;
    int height;
};

bool cmb(plain a, plain b) {
    bool t = false;
    if (a.width < b.width) {
        return true;
    }
    else if (a.width > b.width) {
        return false;
    }
    else {
        return a.height < b.height;
    }
}

int main()
{
    int n = 0;
    cin >> n;
    cin.ignore();
    vector<plain> ps(n);
    vector<int> dp(n, 0);
    for (int i = 0; i < n; i++) {
        cin >> ps[i].width >> ps[i].height;
        cin.ignore();
    }
    sort(ps.begin(), ps.end(), cmb);
    dp[0] = 1;
    for (int i = 1; i < n; i++) {
        for (int j = i - 1; j >= 0; j--) {
            if (ps[j].height <= ps[i].height)
                dp[i] = max(dp[i], dp[j]);
        }
        dp[i]++;
    }
    int m = 0;
    for (int i = 0; i < n; i++) { m = max(m, dp[i]); }
```

```java
import edu.princeton.cs.algs4.WeightedQuickUnionUF;

public class Percolation {
    private final boolean[][] grid; // 网格，表示是否被阻塞  3个用法
    private final int size;         // 网格尺寸  16个用法
    private final WeightedQuickUnionUF frontUf; //  11个用法
    private final int top;  4个用法
    private final int bottom;  3个用法
    private int openSitesNumber;  3个用法

    // creates n-by-n grid, with all sites initially blocked
    public Percolation(int n) {...}

    // opens the site (row, col) if it is not open already
    public void open(int row, int col) {...}

    // is the site (row, col) open?
    public boolean isOpen(int row, int col) {...}

    // is the site (row, col) full?
    public boolean isFull (int row, int col){...}

    // returns the number of open sites
    public int numberOfOpenSites() { return openSitesNumber; }

    // does the system percolate?
    public boolean percolates() { return frontUf.find(top) == frontUf.find(bottom); }

    // test client (optional)
    public static void main(String[] args) {
        Percolation p = new Percolation( n: 3);
        p.open( row: 1,  col: 3);
        p.open( row: 2,  col: 3);
        p.open( row: 3,  col: 3);
        System.out.println("Percolates: " + p.percolates());
    }
}
```



```csharp
using System;
0 个引用
class Program
{
    0 个引用
    static void Main(string[] args)
    {
        string _name = null;
        _name = Console.ReadLine();
        var _str = "Hello, " + _name + ", Meow~";
        Console.WriteLine(_str);
    }
}

// Console.WriteLine("Hello, World!");
// 顶级语句好用喵
4 个引用
public class Student(string name, int age)
{
    private string _name = name;
    private DateTime _createdTime = DateTime.Now;
    0 个引用
    public Student(string name):this(name, 18)
    {

    }
    0 个引用
    public Student(string name, int age, bool flag):this(name, age)
    {

    }
    0 个引用
    public void PrintInformation()
    {
        Console.WriteLine(age);
        Console.WriteLine(_name);
    }
}
```

```python
import random
import numpy as np
import scipy.ndimage
from scipy import ndimage, spatial
import cv2
from os import listdir
import matplotlib.pyplot as plt
IMGDIR = 'Problem2Images'

def gradient_x(img):...

def gradient_y(img):...

def harris_response(img, alpha, win_size):...

def corner_selection(R, thresh, min_dist):...

def histogram_of_gradients(img, pix):...

def feature_matching(img_1, img_2):...

def test_matching():...

def compute_homography(pixels_1, pixels_2):...
def align_pair(pixels_1, pixels_2):...
def stitch_blend(img_1, img_2, est_homo):...
def generate_panorama(ordered_img_seq):...

if __name__ == '__main__':...
```

总而言之

我们会发现，接口的几个显著特点：

- 是系统与系统之间的桥梁；
- 调用者（Caller）不知道被调用者（Callee）内部的运行情况（即黑箱BlackBox）；
- 接口设计存在一定标准，以具有泛用性；

广义定义

- **定义**：接口是不同系统、设备、组件之间交互的**标准化**连接点，定义了通信规则、方法或协议以实现功能调用或数据传递，广泛用于硬件、软件和编程领域。

- 更加广义而抽象的说，接口就是两个系统之间直接交互的媒介，其定义了标准且通用的交互模式，在隐藏内部的同时与外界进行交互。

接口的特征

- 交互性：支持信息、能量、物质的交换；
- 边界性：接口一定位于系统的边缘；
- 规范性：接口需要遵循一定的明确的规则/协议；
- 实现无关性：接口隐藏内部的实现细节；
- 多态性：支持不同实体实现同一接口；
- 最小完备性：接口提供恰好足够的功能；
- 错误容忍性：能够处理交互中的异常；
- 进化性：能够适应需求变化。

什么是API？

- API，Application Programming Interface，即应用程序编程接口。
- API是一种允许不同软件系统之间进行通信和交互的工具。它定义了一组规则和协议，使得不同的应用程序、服务或系统能够共享信息和功能，从而实现互联互通。
- API本质上也是一种接口，一种“合约”，或者说是一种**标准化通信协议**。实际上在编程中我们已经用到了最简单形式的API：STL 和 函数。

举例：

比如说大家最熟悉的 C++，需要打印文字到控制台上，我们调用 <iostream>，并写下：

```
std::cout << “Hello, world!” << std::endl;
```

本质上就是调用了 ostream 类中的 cout 函数。

以及现在大家更加常用的 algorithm、stack、queue、map、math、vector、string 等等一系列 STL，都只是给我们提供了一个实现具体功能的方法，一种接口，而我们并不会关心实现的细节。

市面上的API

- 市面上更多的API有着各种各样的协议，例如WebAPI、RESTful API、gRPC API等等。API的功能足够繁多和强大，只要有需要，就一定可以找到你需要的API，例如天气查询、地理位置、便捷支付、航班/酒店预订、账号登陆……

- API的分类标准及其混乱，我用了10分钟，检索到了API的几十种分类标准[头痛]，当然最常用的还是REST架构的API

一个简单的示例（以python的requests库为例，详细过程下学期的程设会讲）：

```python
import requests  # 导入 requests 库
data = {'key' : 'value'}  # 传入的数据
url = "https://www.example.com" # URL，即我们想要访问的网页
# 也是一种HTTP接口
response = requests.post(url, data=data)
# 发送 POST 请求
print(response.text)
# 打印返回内容
```

-

API vs SDK vs IDE

- **SDK**：Software Development Kit，**软件开发工具包**，相较于API作为通信的连接点，SDK相当于工具箱，配备了 Tools、Libraries、APIs、Documents。一个好的SDK会处理不同平台的复杂性，确保应用在不同设备上运行一致。很多 SDK 都配备了预构建的API客户端（预制API说是），以简化集成过程。SDK负责了身份验证、发送请求、接受相应并转化为可接受的格式。
- 例如：Android SDK、IOS SDK、JDK、Unity、Unreal Engine、谷歌云、亚马逊云等
- **IDE**：Integrated Development Environment，**集成开发环境**，用于提供程序开发环境的应用程序，一般包括代码编辑器、编译器、调试器和图形用户界面等工具。集成了代码编写功能、分析功能、编译功能、调试功能等一体化的开发软件服务套。具备这一特性的软件或者软件套（组）都可以叫集成开发环境。
- 例如：VS、VS Code、IDEA、PyCharm、Dev C++……（我说VSC永远的神，谁赞成谁反对）

API vs SDK vs IDE

- IDE可以一站式地配置SDK和API，以生产APP。
- 用一个比较抽象的比喻：
- API：调味料
- SDK：调味粉包，成分（APIs）+ 使用说明（Documents）
- IDE：餐车，想加什么加什么
- （其实有一点不合理的是，编译器相当于厨具，应该也是包含在SDK内部的）

大模型 API

- 大模型API相较于传统API已经有了显著的区别。
- 传统API是针对特定任务/功能设计的接口，输入、输出非常明确，即严格匹配参数;
- 而大模型API输入是自然语言，输出是生成的文本/代码，具有不确定性，而且具有上下文能力。
- 大模型API绝大多数不是免费的，或者会限制一段时间内传入的token数。

- Tokens，即令牌，是大模型处理文本时的**最小计算单位**，可以理解为语言模型“切分”文本的片段（可能是单词、部分单词或符号）。

API & API Key

- API Key是一种身份验证手段，在购买API服务后，会为你的账户创建唯一的API Key。其用途主要是防止滥用而导致计算量浪费；以及付费大模型的收费标准。永远将API Kay不要硬编码到前端代码中（用环境变量或后端代理，不要学示例代码）。

实践部分

- Cherry Studio / ChatBox
- Roo Code (Cursor)
- Python + OpenAI (*)