C++ 基本语法与 Python 的区别

第一部分：程序的基本结构

1. Python 基础程序 和 C++ 的基础程序

在 Python 中，你想打印一句话，只需要直接写一行代码：

```python
# Python 代码
print("Hello, World!")
```

Python 是**解释型语言**，它从第一行开始直接往下读并执行。

但在 C++ 中，哪怕只是打印一句话，你也需要搭建一个完整的代码框架结构。因为C++ 是**编译型语言**，代码必须先完整翻译成机器码，然后才能运行，以下的代码为 C++ 中 Hello world 程序实现。注：C++代码中 `//` 后的内容表示注释。

```cpp
// C++ 代码的基本框架
#include <iostream>
using namespace std;

int main() {
    cout << "Hello, World!" << endl; // 这是注释，用于输出 Hello, world!
    return 0;
}
```

2. C++ 结构拆解（为什么要有这么多行？）

- `#include <iostream>`：类似于 Python 的 `import`。`iostream` 是"输入输出流"的意思，没有这一行，C++ 连怎么把字显示在屏幕上都不知道，这一行相当于是拿到了 `iostream` 这个工具箱，让它可以获取里面的输出工具。
- `using namespace std;`：声明使用"标准命名空间"。你可以把它理解为告诉 C++："接下来我用到的一些基础工具，都去标准工具箱（std）里找"。
- `int main() { ... }`：**这是 C++ 最重要的地方，叫做"主函数"。**在 C++ 里，程序永远从 `main` 函数的第一行开始执行。大括号 `{}` 里面的内容，就是要执行的具体指令代码。
- `return 0;`：表示程序正常结束，向操作系统汇报圆满完成了任务，没有错误。

3. 两个语法习惯的区别

- **分号 `;` 是结尾**：在 Python 里，换行就代表一句话结束。**在 C++ 里，几乎每一条语句结束时，都必须加上英文分号 `;`。** 忘记写分号是 `python` 使用者在 C++ 里最常犯的错误之一。
- **大括号 `{}` 取代了缩进**：Python 靠严格的缩进来判断代码块。C++ 不管缩进（哪怕你全写在一行它也能运行），C++ 是靠大括号 `{}` 来圈定代码块的。当然，为了人类易读，我们在 C++ 里依然会保持良好的缩进习惯。

第二部分：变量与数据类型

1. 动态类型 和 静态类型

在 Python 里，变量像是一个有弹性的袋子，你往里面装什么，它就是什么：

```
# Python: 动态类型
age = 18        # 此时是整数
age = "十八"    # 突然变成字符串了，Python 完全OK
```

C++ 是严格的"静态类型"语言。在 C++ 里，变量更像是一个定制的容器，你在制造（声明）这个容器的时候，**必须提前规定好它只能装什么类型的东西**，而且永远不能变。

那么**在 C++ 定义变量的语法是** **`数据类型 变量名 = 初始化值;`**（其中 `= 初始化值` 的部分可以省略不进行初始化）。

```
// C++是静态类型（必须先写数据类型，再写变量名字，后面可以进行初始化）
int age = 18;
// age = "十八"; // 错误！C++ 会直接报错：不能把文字塞进整数盒子里！
```

2. C++ 中最常用的基本数据类型

在 Python 里的类型，在 C++ 里都有对应，但划分得更细：

- `int`：整数（例如 `10`，`-5`，`0`）。
- `double`：双精度浮点数，也就是小数（例如 `3.14`，`-0.99`）。C++ 中还有一种叫 `float` 的小数，但 `double` 精度更高，更常用。
- `char`：单个字符。注意！在 C++ 里，**单字符必须用单引号**（例如 `'A'`，`'?'`），双引号是留给字符串的。
- `string`：字符串（例如 `"Hello"`）。需要引入 `#include <string>` 才能使用，并且字符串需要用双引号包裹。
- `bool`：布尔值。代表真假。

3. 变量的声明与初始化

在 C++ 里，你可以先准备好盒子（声明），以后再往里装东西（赋值）：

```
int score;        // 声明变量：准备了一个叫 score 的整数盒子
score = 100;      // 赋值：把 100 放进去

double price = 9.9; // 声明并同时初始化（推荐这种做法）
```

第三部分：输入与输出

在 Python 里，我们用 `print()` 输出，用 `input()` 输入。
在 C++ 里，输入输出的概念有点像是一条河流。

1. 输出：`cout`

C++的输出是使用 `cout` 配合输出符号 `<<` （像两个向左的箭头，表示把数据流向屏幕）。

```
// Python 的输出语法
# age = 16;
# print("我的年龄是", age)

// 对应 C++ 的实现：
int age = 16;
cout << "我的年龄是 " << age << endl;
```

- 你可以用 `<<` 把很多东西串联起来。
- `endl` (End Line) 表示换行，相当于 Python `print` 默认的换行效果。你也可以在字符串里添加 `\n` 来换行。

2. 输入：`cin`

C++的输入是使用 `cin` 配合输入符号 `>>` （箭头向右，表示把从键盘读到的数据流向变量）。

输入这里 C++ 比 Python 方便一点！
在 Python 里，`input()` 拿到的永远是字符串，你要用 `int(input())` 强行转换。
但在 C++ 里，因为变量已经提前规定好了类型，`cin` 会**自动**把你的键盘输入转换成合适的类型！

```
// Python 代码：
# age_str = input("请输入年龄：")
# age = int(age_str)

// C++ 代码：
int age; // 获取输入前，必须先准备好一个空盒子来装这个输入的数据
cout << "请输入年龄："; // 提示用户
cin >> age; // 程序暂停，等待用户输入，并自动转为整数放进 age 盒子
```

第四部分：条件判断

条件判断的逻辑在任何编程语言里都是相通的（无非就是如果...就...否则...），主要的区别在于**符号和格式。**

1. 结构对比（`if - elif - else` vs `if - else if - else`）

我们直接看对比。假设我们要编写根据分数判断 优秀、及格、不及格 的三个等级的代码：

Python 版本：

```
score = 85

if score >= 90:
    print("优秀")
elif score >= 60:
    print("及格")
else:
    print("不及格")
```

C++ 版本：

```cpp
int score = 85;

// 注意 1：条件必须用圆括号 () 包起来！
// 注意 2：执行的代码块必须用大括号 {} 包起来！
if (score >= 90) {
    cout << "优秀" << endl;
}
else if (score >= 60) {  // 注意 3：C++ 里没有 elif，必须写完整的 else if
    cout << "及格" << endl;
}
else {
    cout << "不及格" << endl;
}
```

2. 逻辑运算符的变换 (and, or, not)

在 Python 里，判断多个条件时，使用的是英语化的 `and`、`or`、`not`。
在 C++ 里，需要使用数学领域的符号：

- **与 (and) 变成** `&&`
- **或 (or) 变成** `||`（竖线符号，通常在回车键上方，按 Shift + \ 打出）
- **非 (not) 变成** `!`

举个例子：判断一个人是不是青少年（年龄在 13 到 19 岁之间）：

- Python: `if age >= 13 and age <= 19:` (Python 甚至允许 `if 13 <= age <= 19:`)
- C++: `if (age >= 13 && age <= 19)` (但是C++ **绝对不能**连写，必须用 `&&` 连接两个独立的判断)

3. 布尔值的细微差别

- Python 的真假是首字母大写的：`True` / `False`
- C++ 的真假是全小写的：`true` / `false`

作为有 Python 基础的同学，在写 C++ 时一定要注意以下几个问题：

1. **分号**：每句话代码都要加上 `;`。
2. **类型**：要用一个新变量，必须要规定数据类型，例如 `int`、`double`。
3. **括号**：`if` 后面，条件需要加上 `()`。并且代码块也要加上 `{}`。
4. **结构**：写代码前要写上C++程序框架，`int main() { ... }`，以及前面的 `include` 以及 `using namespace std;`。