陆港第一小学第一节听课笔记

什么是程序

> 程序就是一组计算机能够识别和执行的指令序列，讲白了就是一条条指令，让计算机按照这些指令一步步去操作和执行。那么由我们编写相关的程序并让计算机对程序进行执行的过程就是在编程。

什么是C++

> C++是一种高级语言，由C语言扩展升级而成，是一种较为底层的编程语言，由于其高效的执行效率以及较短的代码量，被广泛应用于竞赛使用。
>
> 计算机程序语言需要将计算机能够听懂的语言（如C++，Python等计算机语言），通过输入设备，输入到计算机中，使计算机理解并按照程序规则输出的语言。

第一个程序——输出（cout）

> 我们在编写一个C++程序的时候，就像是在写作文一样，将我们人类想要表述的内容转化成计算机的语言并输入进**编译器**中，再对代码编译和运行，就可以让计算机执行相关的指令，但是，并不是我们写任何内容计算机都是可以直接识别的，需要一定的语法，以及使用一些工具。
>
> 在我们学习到的第一个C++代码中，我们就需要了解到，想要完成一个程序，需要用到哪些工具。

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4     cout << "Hello World!" << endl;
5     return 0;
6 }
```

这个程序的作用是在计算机的**控制台(Console)**中显示出$Hello\quad Wolrd$（你好，世界）这句话。

我们的程序就是让计算机通过控制台来做到能让现实世界的人看到计算机具体的运算过程的一个窗口。

那么，在我们的程序中有个非常重要的语句，它是**cout**语句，全称为**(Console output)**。控制台输出语句，就如它的名字一样，它的作用就是让计算机把文字或者信息输出在计算机的控制台中，这在我们的第`4`行代码中有体现。

那么同学们还会发现，在`cout`和输出的`Hello World`之间还有一个像是书名号一样的符号`<<`。这是**流运算符**。它像水流的方向一样，把$Hello\quad World$这个东西，通过这个水流，流向了`cout`的`c`，也就是控制台，于是在我们的控制台上就呈现出了文字，那么，紧随其后的下一个语句`endl`，全称是$EndLine$结束当前这一行也就是换行，也进入到了控制台中，所以控制台接下来换了一行。那么，同学们还会发现，我们输出的那句话$Hello\quad World$是用双引号括起来的，这就像是我们文章中的角色想要发言说话的时候需要用双引号将想说的内容括起来一样。在C++中，由双引号括起来的内容被表述为**字符串**，字符串的具体概念会在我们之后的课程中带大家深入了解。

那么程序中其他的部分难道就没有用了吗？

不是这样的，其中第`1`行的

```
1 #include <iostream>
```

这是我们C++当中的**头文件**

第`2`行中的

```
1 using namespace std;
```

这句话也叫**标准命名空间。**

这两句话合起来的作用就是告诉计算机我想使用`cout`这个工具，请打开装有`cout`的工具包吧。那么没有这两句话的加持，计算机也不知道`cout`的含义，也不会执行`cout`应有的作用。

程序的第`3`行

```
1 int main() {
```

这句代码叫做程序的**主函数**，程序的指令也就是从主函数的大括号内部开始执行的，各位同学可以把它看做是程序的入口。

程序的第`5`行

```
1 return 0;
```

它是**返回**的意思，在我们C++代码的主函数中的含义也是结束函数，退出程序的意思，所以大家看到这句代码就意味着程序到此结束了，计算机不会再执行`return 0`后方的代码了。

程序的顺序结构——变量的定义和使用。

数据类型

> 在现实生活中，有非常非常多的数据，构成了我们的生活，比如说年龄是个**整数**，商品的价格是个**小数**，汽车车牌号的省份地区第二位是个**字符**，火车票的列车号是个**字符串**等等等等，在计算机中，如果我们不规范数据的类型，这样就导致计算的混乱，比如说我们拿$Hello \quad World$去做乘法，或者是用小数减去一个字母。所以我们要严格规定每种数据的类型。

以下是常见数据的类型：

程序中的数据类型	中文名称	举例
int	整型	73
double	双精度浮点型(小数)	3.1415
char	字符型	'A'
long long	长整型	9223372036854775807
string	字符串型	"Hello World"

变量的定义

> 计算机之所以能够计算数值，是因为它可以存储一些值，可能是永久存储，可能是临时存储，比如说我们想算5 + 8的和，那么人脑也是通过先记住了这两个数，然后再对这两个数进行求和的过程。对于计算机也一样，所以我们需要一些能够保存值的东西，我们给他起个名字叫做**变量(Variable)**

想要定义一个变量需要知道两个必要的条件和一个不必要的条件

两个必要条件：

- 变量的数据类型
- 变量名

一个不必要的条件：

- 变量的初始值

那么定义一个变量需要遵循如下的原则：

```
1) 数据类型 变量名;
int a;
2) 数据类型 变量名 = 变量的初始值
int a = 114;
```

同学们需要注意的是，`=`在我们的C++当中并不是等号，而是被称为赋值号，它的作用是**将赋值号右边的值赋给左边的变量**，所以我们知道，等号的左边必须要是一个变量，等号的右边可以是一个数值，可以是另外一个变量，变量一旦被赋了值后，我们再提到它时，就会被它所赋的值代替。

比如说我们可以写出如下的代码。

```
#include <iostream>
using namespace std;
int main() {
    int a = 114;
    cout << a;
    return 0;
}
```

首先，大家需要先明白一件事，在我们的C++语言中，程序的执行是有先后顺序的，这是基本的顺序结构，刚刚我们也 了，主函数是程序的入口，那么大家可以想象有一个箭头，一开始指向了上方这个程序的第`4`行，每执行完一句话后，遇到一个分号后，箭头指向下一行。

在这个程序的第`5`行中，我们输出了变量`a`，其实也就是输出变量`a`的值，那么在这个程序的第`4`行中，我们定义了一个变量叫做`a`，它的数据类型是`int`，所以可以保存整数，并且我们定义它的值为`114`，所以我们输出变量`a`其实也就是输出了`114`。

标识符命名的规范

那么同学们此时可能会思考一个问题：是不是我给变量起任何名字都是可以的呢？它可不可以不叫`a`？可以不可以是一个单词？

那么变量名是有统一规范的，也叫标识符命名规则：

只允许出现数字、字母、下划线，但是不能以数字开头。

也就是说不能够出现`26`个大写字母,`26`个小写字母，`0-9`十个数字，以及下划线以外的内容了，而且不允许是数字开头，只要遵循这个原则，写出的变量名都是符合规范的，比如：`____Shield`，`Apple300`，`get_Ready`。

算术运算符

> 在第一节课（2024年3月22号）的学习中，我们在课堂上只了解了加法运算符，在后面的课程中，我们会了解更多的运算符知识。

> 在我们的数学当中，学习了+ -× ÷四种运算规则，在C++的代码当中，他们被称为：算术运算符**，不过相信有稍微对电脑有些了解的同学也比较清楚，计算机的键盘当中并没有我们数学当中的乘号(`×`)和除号(`÷`)。

在计算机中的四则运算被写作

```
int a = 1, b = 2;
cout << a + b << endl;//加法运算符
cout << a - b << endl;//减法运算符
cout << a * b << endl;//乘法运算符
cout << a / b << endl;//除法运算符
```

其中乘号被替换成了数字键`8`上方的这个星号(`*`)，除号被替换成了`M`键右边的右边的右边的这个左斜杠符号，这个除号在C++中被称为整除，不保留任何小数部分的除法，比如说：现在我们想要计算$3\div2=1.5$，在计算机中写作$3/2=1$所有的小数部分全部舍去。

所以刚刚的代码各位同学可以加上开头结尾，发现最后在计算a/b的结果应该是`0`，因为$1\div2=0.5,1/2=0$。

那么，既然整除符号无法完成我们计算小数的要求，在很多特定情况下，我们写代码是为了能够计算小数相关的时候该如何处理呢?

还记得刚刚提到的**数据类型**吗？在我们的数据类型当中有一个叫做`double`的数据类型，这个数据类型可以让我们能够保留小数。

那么同学们可以尝试写出以下的代码

```
#include <iostream>
using namespace std;
int main() {
    int a = 1, b = 2;
    double c = a / b;
    cout << c << endl;
    return 0;
}
```

各位同学觉得答案应该是`0.5`还是`0`呢?

奇怪的事情发生了，结果还是`0`，因为**赋值号左右两边的类型不匹配。**各位同学观察一下我们的第`5`行代码，赋值号的左边是`double`类型的变量`c`，赋值号右边是两个整型变量的除法，所以计算机会认为右边是整型与整型相运算，结果就也为整型，所以右边的结果赋值给到了变量`c`，最终也是以整型的形式保存下来的，并没有改变。

那么想要成功的计算出`double`类型的小数，我们需要做什么修改？是不是得让赋值号的右边也是一个`double`类型的结果。在这里有两种修改方式。

方法一：因为赋值号右边是两个`int`的数值相计算，我们只需要把其中一个`int`类型的变量修改为`double`，或者将所有变量全部修改为`double`，计算机就不会认为结果也应该是`int`，而是以`double`存储，这样赋值号左右两边类型相匹配，能够获得答案。

方法二：我们可以在给变量`c`赋值的时候写：`double c = 1.0 / 2`此时计算机会发生一个叫做**隐式转换**的过程，右边的$1.0/2$中包含了有小数部分的**1.0**，计算机就会认为这个整体应该用`double`来存储这样赋值号的左右两端类型相匹配。也能计算出结果`0.5`。