# 《变量与数据类型2》课堂笔记

## 1. 详解数据类型

### 1.1 整型（存储整数）

- 长整型 `long long`：可存储的数字范围是`-2147483648 ~ 2147483647`(-21亿～21亿，10位数，2e9)
- 整型 `int`：可存储的数字范围是`-9223372036854775808 ~ 9223372036854775807`（19位数，9e18）
- 短整型 `short`：可存储的数字范围是`-32768 ~ 32767`

### 1.2 浮点型（存储小数）

- 单精度浮点型`float`：整个数的有效数字6～7位
- 双精度浮点型`double`：整个数的有效数字15～16位

### 1.3 字符型（存储字符）

C++中要表示文字，有字符和字符串两种形式

- **字符型`char`**：使用单引号扩起来的单个字符。比如`'A'`、`' '`、`'!'`、`'#'`
- **字符串型`long long`**：使用双引号扩起来的单个或多个字符。比如`"A"`、`"Hello, World!"`、`"Tony"`

字符在计算机中是以数字的形式存储的，人们在给字符编码时遵循的是`ASCII`>标准。

> 字符`'A'`的ACSII码是`65`。
>
> 每一个字符在计算机中都有唯一对应的编码，类似于我们的身份证号。
>
> 所有的字符型数据都可以看作是特殊的整型数据，字符型变量可以与整数进行赋值，也可以参与运算。

例：输出这个字符'A'加上数字3的结果

```
#include <iostream>
using namespace std;
int main(){
  char a = 'A';
  cout << a + 3;
  return 0;
}
```

输出结果：`68`

> 因为a是字符char，与整型混合运算时会转换为数字65参与运算

若想输出68对应的字符，可以这样写：

```
#include <iostream>
using namespace std;
int main(){
  char a = 'A';
  cout << (char)(a + 3);
  return 0;
}
```

或：

```
#include <iostream>
using namespace std;
int main(){
  char a = 'A';
  a = a + 3;
  cout << a;
  return 0;
}
```

> a + 3的结果是68，赋值到字符型变量a时被强制变为char类型，所以输出了D

## 2. 隐式转换

示例：金字塔的底是正方形，侧面由四个大小相等的等腰三角形构成。试编一程序，输入底和高，输出三角形的面积。（提示：三角形的面积公式是面积s = 底a * 高h /2）

编程：

```
#include <iostream>
using namespace std;
int main(){
  int a, h, s;
  cin >> a >> h;
  s = a * h / 2;
  cout << "s = " << s << endl;
  return 0;
}
```

程序运行结果：

`s = 7`

> 三角形 s = a * h / 2，应该是7.5才对！程序为什么输出7，而不是7.5呢?

解析：

- 计算机在计算`a * h / 2`时，先找到变量`a`、`h`的值，由于`a`、`h`都是整型，所以`a * h`的结果也是整型，

值为`15`

- 然后计算`15 / 2`，由于`15`和`2`都是整型，所以`15 / 2`的结果也是整型，`15`与`2`做整除运算，所以结果是`7`。`7`存入整型变量`s`中。

> 整除：自动取整，只取商的部分，小数（余数）部分截去
>
> 例：17/2 的结果是 8，99/2 的结果是 49

- 所以只要我们在`a * h / 2`这条语句中改变一下`2`的数据类型，就可以了。

改进编程：

```
#include <iostream>
using namespace std;
int main(){
  int a, h, s;
  cin >> a >> h;
  s = a * h / 2.0;     // 将之前的除数2改为浮点数2.0
  cout << "s = " << s << endl;
  return 0;
}
```

程序运行结果：

```
s = 7.5
```

#### 2.1 隐式转换规则

像刚才的题目中，在一个算术表达式`15/2.0`中出现两种以上的数据一起进行混合运算时，数据会按照规则往同一种数据进行转换，转换为同一种数据后再进行计算。

**隐式转换规则：**从存储范围较小的类型转换为存储范围较大的类型

> char < short < int < long long < float < double

## 3. 强制转换

#### 3.1 强制转换的定义

有时候，我们想要主动将数据从一种类型转换为另一种类型，这时我们需要用到强制转换。

> 任意情况下都可以使用强制转换

例：我们有一个小数，想转换为整数输出（向下取整）

编程：

```
#include <iostream>
using namespace std;
int main(){
  double f = 3.74159;
  cout << (int)f; // 将浮点数f转换为整数输出
}
```

输出结果：

`3`

> 用int强制转换后的数据是向零取整的，也就是不考虑四舍五入，直接把小数点后的数据抹掉

### 3.2 强制转换的三种写法

- **(数据类型) 变量名**：`(int)a`
- **数据类型(变量名)**：`int(a)`
- **(数据类型) (变量名)**：`(int)(a)`

> 三种语法没有差别，都可以使用

- 还有一种强制转换的方法是将想要转换的数据赋值给对应的变量，也可以进行转换

```
double f = 3.14159;
int n;
n = f;  // 将浮点数f强制转换为整数
```

## 4. 课中练习

问题描述：输入摄氏温度值C，求对应的华氏温度值F并输出

程序输入：对应的摄氏温度值C

程序输出：对应的华氏温度值F

提示：F = C * 9 / 5 + 32，温度值取浮点数类型

> 思路分析：
>
> - 需要两个变量，分别存放摄氏度C和华氏度F，且变量类型是浮点型（题目没说就默认定义为双精度浮点型）
> - 输入一个数据，代表摄氏度，所以cin输入到变量C中
> - 输出一个数据，代表华氏度，需要先计算华氏度的值：`F = C * 9 / 5 + 32`可以将计算后的结果放到变量F中，然后cout输出

编程：

```
#include <iostream>
using namespace std;
int main(){
  double C, F;
  cin >> C;
  F = C * 9 / 5 + 32; // 由于C是浮点型，混合运算时与它运算的数据会转换为浮点型，所以不需要改其他数为小数
  cout << F;
  return 0;
}
```

问题描述：输入一个字符，输出它的ASCII值

> 思路分析：
>
> - 需要一个变量`ch`，存放字符，所以变量数据类型是字符型
> - 输入一个字符数据到变量`ch`中
> - 需要以整数类型输出，强制转换为int再输出

编程：

```
#include <iostream>
using namespace std;
int main(){
  char ch;
  cin >> ch;
  cout << (int)ch;
  return 0;
}
```

问题描述：输入一个ASCII值，输出对应的字符

> 思路分析：
>
> - ASCII值是个整数，需要一个整型变量`n`来存放
> - 输入一个整数数据到变量`n`中
> - 需要以字符类型输出，强制转换为char再输出

编程：

```cpp
#include <iostream>
using namespace std;
int main(){
  int n;
  cin >> n;
  cout << (char)n;
  return 0;
}
```