程序设计语言与方法(C语言)

第三章 算术运算与表达式

简单的算术运算符

- 简单的算术运算符号
  - 一元运算
    - 负-
  - 二元运算
    - 加+、减-、乘*、除/、模（求余数）%、

除法运算

- 结果与数据类型相关
  - 整数相除的结果是商
    - 1 / 2 的结果是0，而不是0.5
      - int x = 5, y = 3; x / y 的结果是?
  - 实数相除的结果是实数
    - 1.0 / 2、1 / 2.0、1.0 / 2.0的结果都是0.5
    - double x = 4.0, y = 2.0; x /y 的结果是?

模（求余）运算

- 只对整型数据进行模运算
  - 3 % 5、11 % 2、100 % 10
  - int x = 123;
    - x / 100 的结果是什么?
    - x % 100 的结果是什么?
- %运算的操作数可以是实型数么?
  - 与编译器相关
    - 不允许时，编译器会报错!
    - 允许时，是先截尾（取整），再做运算，如： 11.8 % 2的结果是5

模运算余数的符号

- 与被除数的符号相同
  - 5 % 2 和 5 % -2 的结果都是正数
  - -5 % 2 和 -5 % -2 的结果都是负数
  - 5 / -2 = ?
  - -5 / 2 = ?

混合运算与算符优先级关系

- 取反运算（负号）的优先级最高
- 乘*、除/和模%次之，但它们三个的优先级相同
- 加+、减-的优先级最低，但它们俩的相同
- 使用括号改变它们的运算次序
- 混合运算过程
  - 取反是右结合
  - 自左向右按优先级运算

-3 * 2 - 1 + 3的运算过程是?

3 * -2 - 1 + 3的运算过程是?

不同类型数据的混合运算

- 2 + 3.14 * 3 * 3
- 2 + 1.5 - 1.5
- 2.4 + 5 + 7 * -1.3
- 结果均为实型数据！！

复合的赋值运算符

- 简单算术运算符与赋值符号的组合
  - +=、-=、*=、/=、%=
  - A @= B; ==〉 A = A @ B;
- 示例
  - n *= m; n *= m + 1;
  - a中的数据为3时，经下列语句后，a中的数据是多少？
    - a += a -= a * a;
    - a += a -= a *= a;

通常不建议使用，
连续的复合赋值语句！

增1(++)和减1(--)运算

书写简洁，且直接与机器指令相对应

执行效率高！

- 前缀
  - ++x; => x = x +1;
  - y = ++x;
    - x = x + 1; y = x;

先运算、再取值

- 后缀
  - x++ => x = x + 1;
  - y = x++;
    - y = x; x = x + 1;

运算符的前/后只能是变量名！

先取值、再运算

再说常量

- 幻数（直接常量）
  - 可能会有什么问题？ P34
- 宏常量和宏替换（符号常数）
- const常量
  - 不允许变量其值的变量
  - 定义时必须给定值

宏常量与宏替换

- 给常数一个名字
  - #define PI 3.14159
- 编辑程序（程序员）
  - 直接使用名字来表示一个常数：s = PI * 3.0 * 3.0;
- 编译程序（机器）
  - 宏替换：PI => 3.14159
  - 先用3.14159替换符号PI，再进行后续处理

宏常量示例

预处理命令/预编译指令

```
#include <stdio.h>
/* 宏定义：PI是宏名 */
#define PI 3.14159
int main()
{
  double r = 3.0;
  printf("s = %f”, PI * r * r);
}
```

宏替换

==========

```
#include <stdio.h>
int main()
{
  double r = 3.0;
  printf("s = %f”, 3.14159 * r * r);
}
```

将宏名替换成宏定义中的字符串

#define 宏名 字符串

宏替换实质上就是字符串替换，不做任何语法检查

const常量

- 形式上是一个变量，但不允许其值发生变化
- 有明确的数据类型
- 有名字（标识符）
- 有固定的存储位置
  - const double PI=3.14159;
  - y = PI * 3.0 * 3.0;

const 常量示例

```
#include <stdio.h>
/* const常量声明语句 */
const double PI = 3.14159;
int main()
{
  double r = 3.0;
  printf("s = %f”, PI * r * r);
}
```

```
#include <stdio.h>
/* 宏定义：PI是宏名 */
#define PI 3.14159
int main()
{
  double r = 3.0;
  printf("s = %f”, PI * r * r);
}
```

const常量与宏常量的异同是什么?

表达式与表达式的值

- 表达式
  - 单个的常量、变量、函数都是一个表达式
    - 函数：sqrt、pow等系统提供的或自定义的计算过程
  - 使用算符连接的常量、变量、函数等混合运算的式子（算符不能省略）
    - #define PI 3.14159
    - PI * sqrt(r); /* πr^2 */
- 表达式的值
  - 常量、变量的值、函数的执行结果、混合运算式的计算结果

使用函数

站在别人的肩膀上前进，：）

- 函数：预先定义的一个计算过程（子过程/子程序等）
- 构成要素：返回值 函数名称(形式参数列表)
  - 返回值：计算结果
    - 函数名称：标识符，应用范围唯一
    - 形式参数列表：交由函数计算/或使用的数据序列，可无，有2个或2个以上的参数时，参数间以逗号分隔。
- 使用：函数调用
  - 函数名(实参列表) /* 与形参一一对应 */

了解待用的函数

- 先查阅可完成功能需求（开平方）的函数的声明
  - 头文件：声明/定义了函数的文件
    - math.h
  - 函数的要素：double sqrt(double _x)
    - sqrt
    - 一个形参：_x（不小于0的双精度实型数）
    - x的平方根，双精度实型数

函数调用

- 包含头文件 math.h
  - #include <math.h>
- 准备:声明变量（传递参数和返回值）
  - double x, y; /* 不需要存储返回值时，无需定义 */
  - x = 9.0;
- 函数调用
  - y = sqrt(x); /* 赋值语句 */
  - y = x * sqrt(x) + 4; /* 在表达式中使用函数 */
  - sqrt(x); /* 函数调用语句，调用函数完成功能，但不使用函数的计算结果 */

函数调用示例

```
#include <stdio.h>
#include <mathg.h>

int main()
{
  double a, b ,c s, area;
  printf("Input a, b, c: ");
  scanf("%f, %f, %f", &a, &b, &c);
  s = (a + b + c) / 2;
  area = sqrt(s * (s - a) * (s - b) * (s - c));
  printf("area = %lf\n", area);
}
```

常用函数

- 字符测试函数
- 字符串操作
- 内存管理函数
- 日期与时间函数
- 数学函数
- 文件操作函数
- 进程管理函数
- 文件权限控制
- 信号处理函数
- 接口处理函数
- 环境变量函数
- 终端控制函数

数学函数

函数名称	参数	返回值	调用示例	说明
pow	double x, double y	double	pow(x,y)	求x的y次幂
sqrt	double x	double	sqrt(x)	求x的平方根
abs	int x	int	abs(x)	求x的绝对值
ceil	double x	double	ceil(x)	求x的下限
log	double x	double	log(x)	求x以e为底的对数
sin	double x	double	sin(x)	求x的正弦值
……				

表达式的值的数据类型

- 如果表达式中有不同类型的数据参与运算呢？
  - 1.0 / 5、int x; float y; y / x

统一参与运算的数据的数据类型！
实型常数是double，整型常数是int

- 自动/隐式类型转换

  - 编译器的处理：类型提升（无须逐步提升）
- 强制/显式类型转换
  - 程序员的处理：明确指定待使用数据的数据类型，若还不一致，编译器会再次进行自动转换
  - (数据类型名称)表达式

再看赋值语句

- 变量名 = 表达式;
  - 1. 计算表达式的值
  - 2. 将计算结果赋值给变量
    - 装入由变量名表示的盒子（存储块，若干连续的存储单元）
- 赋值中的类型转换（自动、强制）
  - 自动：将右侧的转换成左侧的，这里不是类型提升!
  - 强制：程序员指定，若还不一致，自动处理

数据损失问题

```
#include <stdio.h>

int main()
{
  int n = 256;
  float m = 3.6;
  double k = 2.4;
  n = m;
  printf("n = %d", n); /* n的值会是多少呢?  */
  k = m;
  printf("k = %lf", k); /* k的值会是多少呢?  */
  k = n;
  printf("k = %d=lf", k); /* k的值会是多少呢?  */
}
```

一些特殊的运算与表达式

- 赋值运算符、赋值表达式与赋值语句
  - x = y = 2; /* 赋值语句 */
    - /* y = 2 是赋值表达式，其中的 = 是赋值运算 */
  - a = b = c = 5.2;
- 逗号运算符、逗号表达式、逗号表达式的值（最右一个）
  - a = 1, 2, 3;
  - a = (1, 2, 3);

回顾：定义/声明变量

- 目的是为了使用数据
- 选择合适的数据类型
  - 包容原则
- 定义一个有意义的名字
  - 变量
    - 实质上是可以放置数据的盒子（存储块）
    - 值是盒子中放的数据
  - const 常量
    - 实质上是一个变量，但其存放的数据固定不变

回顾：运算与表达式

- 运算符号
- 算符优先级
- 表达式：算符与数据组合而成的运算式
- 表达式的值
- 表达式的值的数据类型
- 参与运算的数据的类型转换问题（类型提升）

回顾：赋值

- 赋值符号
- 赋值表达式 x = 3
- 赋值语句x = 3;
- 类型转换
  - 右边向左边转换
  - 不做类型提升，注意：可能会造成数据损失！