C语言入门（*CS100*）Part Two

> *-- By Yiming Li*

> 本文章的内容参考至*Stephen Prata*的《C Primer Plus》

> 本内容适用于*CS*专业学生

运算符

算术运算符：

包括+,-,*,/.

- 但是我们要注意的就是除法

```
printf("%d",10/3);//3
```

如果是整数的计算的话结果一定是整数，如果是小数也会将它的小数部分省略。

- 如果要实现小数的计算：

```
printf("%lf",3.0+10.1);//13.1000
printf("%f",3.0f+10.1f);//13.1000
```

- 同样的如果是小数的计算结果一定是小数：

```
printf("%lf\n",10.+3.0);
```

- 整数和小数的计算结果一定是一个小数

```
printf("%lf\n",10/3.0);//3.33333
```

- 如果是小数之间参与计算的话就可能出现误差。

```
printf("%lf",3.333333333333+6.66666666666);//10.000000
printf("%.20lf",10.0/3.0);//在一定位数的时候会出现不精确的情况
```

计算的时候的一些注意事项：

1. 除法： 不能够除以零
2. 取余：
   1. 运算的数据全部都必须要是整数。`printf("%d\n",10 % 3);`如果是`10.0`就不行。
   2. 获取余数的正负，和第一个数字保持一致

```c
printf("%d\n",10%3);//1
printf("%d\n",-10/3);//-1
printf("%d\n",10%-3);//1
```

3. 和除法一样，除数不能够是0
4. 类型的变换：如果是两个相同类型的数据进行加减运算结果还是一样的类型，但是如果是两个不同类型的数据进行计算，那么在计算前都要进行类型转换。

类型的转换：

1. 隐式转换：将一个取值范围小的，转换成取值范围大的。

```c
short a = 10;
int b = a;
```

在C语言中取值范围的大小关系如下：`double>float>long long>long>int>short>char`

- `short`类型和`char`类型在运算的时候先提升为`int`类型
- 如果在计算的时候存在取值范围更大的时候就会转换成为取值范围更大的部分。
- 在输出的时候如果格式化输出相关内容也会进行隐式转化。

2. 强制转化：目标数据类型变量名 = （目标数据类型）被强转的数据。**有可能会导致数据会发生错误**

```c
int b = 10;
short i = (short)b;
```

自增自减运算符

```c
int a = 10;
a = a + 1;//a++; ++a;
```

注意要单独写成一行才能够成立.

同理还有`a--;--a;`

```c
int a = 10;
int b = a++;
//先用后加
```

```c
int a = 10;
int b = ++a;
//先加后用
```

再来看这种复合的情况：

```c
#include <stdio.h>
int main()
{
    int i = 10;
    int j = 5;
    int k = i++ + ++i - --j - i--;

    printf("%d\n",k);
    return 0;
}
```

- 代码解释

1. `i++` (后置自增)：
    - 使用当前值 `i = 10`。
    - 然后将 `i` 增加到 11。
2. `++i` (前置自增)：
    - 先将 `i` 增加到 12。
    - 使用新的值 `i = 12`。
3. `--j` (前置自减)：
    - 先将 `j` 减少到 4。
    - 使用新的值 `j = 4`。
4. `i--` (后置自减)：
    - 使用当前值 `i = 12`。
    - 然后将 `i` 减少到 11。

- 计算过程 现在我们可以计算表达式的值：
- `i++` 返回 10，然后 `i` 变为 11。
- `++i` 将 `i` 增加到 12 并返回 12。
- `--j` 将 `j` 减少到 4 并返回 4。
- `i--` 返回 12，然后 `i` 减少到 11。

因此，表达式变为：

```
k = 10 + 12 - 4 - 12;
```

简化后：

```
k = 6;
```

- 最终结果
- `printf("%d\n", k);` 输出 `6`。
- `return 0;` 结束程序。

赋值运算符

和python一样我们有：`a = b;a += b;a -= b;a *= b`…

关系运算符

和python一样的：

```
#include <stdio.h>
int main()
{
    int a = 10, b = 11;
    printf("%d\n",a==b);//0
}
```

引入_Bool类型：

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

int main()
{
    int a = 10;
    int b = 11;
    bool isEqual = (a == b);
    bool isLessThan = (a < b);
    bool isGreaterThan = (a > b);

    printf("isEqual: %d\n", isEqual);             // 输出 0 (false)
    printf("isLessThan: %d\n", isLessThan);       // 输出 1 (true)
    printf("isGreaterThan: %d\n", isGreaterThan); // 输出 0 (false)

    return 0;
}
```

逻辑运算符

与&;或||;非!

```
#include <stdio.h>
int main()
{
    // 1:true;0:false
    printf("%d\n", 1 && 0);//0
    printf("%d\n", 1 || 0);//1
    printf("%d\n", !1);//0
}
```

逻辑运算符的短路效果 如果在逻辑判断&/||的时候仅仅通过一边就能够判断整个式子的逻辑的真假的时候，就会导致后面的内容不进行判断就直接输出结果

三元表达式

格式：关系表达式? 表达式1: 表达式2

```
    int a = 10;
    int b = 20;
    printf("%d",a > b ? a : b);
```

运算符的优先级

- 总则：
  - 小括号优先于所有
  - 一元运算符>二元>三元表达式
  - &&>||>赋值运算符

流程控制语句

顺序结构

从上往下顺序进行，是程序的默认的运行顺序

分支结构

if 语句

```
if (关系表达式)
{
  语句体;
}
```

- 几个小细节

1. 在C语言中如果是`if (非0数字)`就表示永远成立，如果是`if (0)`就是表示一定不成立的意思
2. 在大括号中如果语句只有一**行**就可以不用使用大括号。

```
if (关系表达式)
{
  语句体A;
}
else
{
  语句体B;
}
```

```
if (关系表达式A)
{
  A;
}
else if (B)
{
  B;
}
else if (C)
{
  C;
}
else
{
  D;
}
```

`switch`语句

```
switch (表达式)
{
  case 值1:
    语句体A;
    break;
  case 值2:
    语句体B;
    break;
  default:
    语句体n;
    break;
}
```

- 代码解释首先计算机会计算switch语句中的计算的结果，然后会依次与case中的数值进行比较。
- 如果匹配则会执行case n:后面的内容
- 如果和所有的case中的数值都不匹配，则会执行default中的内容。
- 几个细节：
    1. switch表达式最后的计算的结果只能是**字符或者是整数**（其他的不允许）
    2. 同样的case中的内容也只能是字符或者是整数。
    3. case中的数值不能够重复
    4. break表示中端，结束switch语句
    5. default可以写在switch中的任意位置，甚至可以省略不写。
- switch和if的对比：switch只能用于有限个的选项之中，一般是四个case以内。但是switch的执行效率更快（if是从上到下依次匹配$O(0)$，switch是直接定位进行匹配$O(1)$）。

```
int number = 2;
switch (number)
{
  case 1:
    printf("Number is 1\n");
    break;
  case 2:
    printf("Number is 2\n");
    //break;
  default:
    printf("Number is not 1 or 2\n");
    break;
}
```

```
Number is 2
Number is not 1 or 2
```

- 这个叫做case穿透：

1. 还是根据小括号中表达的结果去匹配case
2. 执行对应的case
3. 如果在执行的过程中，遇到了break，那么会直接结束整个switch，如果没有break，那么就会一直执行直到遇到下一个break或者是运行完整个switch
4. case穿透只会往下穿透，不会往上走。

```
// 使用case穿透的功能来完成的程序
#include <stdio.h>
int main()
{
    int month;
    scanf("%d", &month);
    switch (month)
    {
    case 3:
```

```
        case 4:
        case 5:
            printf("spring\n");
            break;
        case 6:
        case 7:
        case 8:
            printf("summer\n");
            break;
        case 9:
        case 10:
        case 11:
            printf("autumn\n");
            break;
        case 12:
        case 1:
        case 2:
            printf("winter\n");
            break;
        }
        return 0;
    }
```

循环结构

- 存在三个内容：

1. 初始化语句
2. 条件判断语句
3. 条件控制语句

for循环：

```
for (初始化语句;条件判断语句;条件控制语句)
{
  循环体语句;
}
/*
for (int i = 1; i<=100; i++)
{
  printf("i = %d",i);
}
*/
```

- for循环的执行流程

1. 执行初始化语句
2. 执行条件判断语句
3. 如果成立就会执行循环体语句；如果不成立就会结束循环

```c
// 例题：输入两个数字输出两个数字中的能够被6和8整除的数字以及个数
#include <stdio.h>
int main()
{
    int a, b, c = 0, num_max, num_min;
    scanf("%d %d", &a, &b);
    num_max = a > b ? a : b;
    num_min = a < b ? a : b;
    for (int i = num_min; i <= num_max; i++)
    {
        if (i % 6 == 0 && i % 8 == 0)
        {
            c++;
            printf("%d ", i);
        }
    }
    printf("\n%d", c);
    return 0;
}
```

while循环

```
while(条件判断语句)
{
  循环体语句;
  条件控制语句;
}
```

- 提问？for和while之间的区别？
- 相同点:运行规则是一样的
- 不同点:
    - for循环中的i是定义在循环中的，是不能够在大括号外中使用
    - while循环中的i是定义在循环外侧的，是可以在大括号外使用的。

```c
// 小练习2：判断回文数
#include <stdio.h>
int main()
{
    long long int a, b, c = 0, original_a;
    scanf("%lld", &a);
    original_a = a;
    while (a / 10LL != 0)
    {
        b = a % 10LL;
        c = c * 10LL + b;
        a /= 10LL;
```

```
    }
    c = c * 10LL + a;
    printf("%lld = c;%lld = original_a\n", c, original_a);
    if (original_a == c)
    {
        printf("Yes\n");
    }
    else
    {
        printf("No\n");
    }
    return 0;
}
```

do···while循环（主要是在选择题中）

```
初始化语句;
do{
  循环体;
  条件控制语句;
}while(条件判断语句);
```

- 执行顺序：

1. 初始化语句
2. 循环体语句
3. 条件控制语句
4. 条件判断语句（如果满足条件就有回到循环体语句）
5. 如果不满足就直接退出循环

```
#include <stdio.h>
int main()
{
  int i = 10;
  do{
    printf("%d\n",i);
    i++;
  }while(i<=5);
  return 0;
}
```

```
10
```

循环结构的高阶使用

无限循环

```
for ( ; ; )
{
  printf("无限循环");
}
```

```
while(1)
{
  printf("无限循环");
}
```

```
do
{
  printf("无限循环");
}while(1);
```

跳转控制语句

- break表示跳出循环
- continue表示跳出本次循环进行下一次循环

循环嵌套

```
#include <stdio.h>
int main()
{
    for (int i = 0; i < 3; i++)
    {
        for (int j = 1; j <= 5; j++)
        {
            printf("excuation of inner loop%d\n", j);
            break;
        }
        printf("end of inner loop\n");
        printf("---------\n");
    }
    printf("end of outside loop\n");
    return 0;
}
```

```
excuation of inner loop1
end of inner loop
---------
excuation of inner loop1
end of inner loop
---------
excuation of inner loop1
end of inner loop
---------
end of outside loop
```

- 可见如果是使用break关键词就会当前的循环（不能够跳出多层循环）

goto（标号）关键词的学习

```
// 在程序的任意位置标号：
a:
  a标号的对应语句

goto a; //表示又跳转到a位置进行程序的进行（包括a:那一行）
```

- 注意上面这个代码是一个死循环(不能够使用break来推出循环error: break statement not within loop or switch)