C Language

Jianfeng He

第一章　C 程序设计初步

§1.1　C 程序的结构和处理过程

§1.2　C 语言的基本数据类型

1.2.1　`char`

转义码表见表 1.1:

表 1.1: escape char

转义字符	意义	功能解释
`\0`	NULL	字符串结束符
`\b`	退格	将光标左移一个字符
`\n`	换行	光标移向下一行的开始
`\f`	换页	(打印机) 换到下一页
`\t`	水平制表符	把光标移到下一个制表位
`\\`	反斜杠	输出反斜杠
`\"`	双引号	输出双引号
`\'`	单引号	输出单引号
`\a`	响铃	警报响铃
`\<num><num><num>`	ASCII 字符	输出八进制数字对应的 ASCII 字符
`\x<num><num>`	ASCII 字符	输出十六进制数字代表的 ASCII 字符

Notation :

1). `printf('%c', '\101')` 与 `printf('%c', 101)` 的输出值分别是 `A` 与 `e`. 这是由于前者按八进制获取 ASCII 字符, 后者进行了隐式的转义, 按照十进制获取 ASCII 字符.

2). ASCII 编码中, 数字 0 ∼ 9 对应的编码是 48 ∼ 47; 大写字母 A ∼ Z 对应的编码是 65 ∼ 90; 小写字母 a ∼ z 对应的编码是 97 ∼ 122

§1.3　输入与输出

1.3.1　格式化输出: `printf()`

1. 简单输出

```
1 printf("<string>");
```

这条语句将直接输出字符串 `<string>`

2. 基本语法

```
printf(%[-][*][m.n][l/h]<format_charactor>, <var_1>, <var_2>, ...);
```

其中各项的意义为:

1). `%`: 占位符, 一个变量对应一个占位符, 该占位符后面的那些字符表示对该变量的格式控制.

2). `<format_charactor>`: 控制数据输出的模式, 常用的格式输出符有见表 1.2

表 1.2: options of format charactor

format charactor	meaning
`d`	decimal, 以 10 进制形式输出带符号整数, 正数不输出负号
`o`	octal, 以 8 进制形式输出无符号整数, 无前缀 `o`
`x`	hexadecimal, 以 16 进制形式输出无符号整数, 无前缀 `ox`
`u`	unsign, 以 10 进制输出无符号整数
`f`	float, 以小数形式输出单、双精度实数
`e`	exponential, 以指数形式输出单、双精度实数
`g`	以 `%f` 或 `%e` 中较短的输出宽度输出
`c`	char, 输出单个字符
`s`	string, 输出字符串

3). `l/h`: 长度修正符, 用于指定对应位置输出数据是按长类型数据输出还是按短类型数据输出.

4). `m,n`: 域宽可选项, 用于指定对应输出项所占的输出宽度, 即指定用多少字符位置来显示对应输出数据.

5). `*`: 对应输出表列中两个连续数据项, 其意义是用前一个数据项的值作为后一个数据项输出的指定域宽.

6). `-`: 减号可选项, 用于指定对应输出数据的对齐方向. 当选用减号时, 输出数据左对齐, 否则右对齐.

Notation :

1). 若 `a` 为 `int` 类型的数据, 则表达式 `printf("%c", a)` 将会输出 `a` 对应的 ASCII 码值.

§1.4　基本运算符和表达式运算

1.4.1　自增与自减运算符

自增运算符 (`++`) 与自减运算符 (`--`) 用以对变量进行增量赋值. 允许的语法有:

```
++<var_name>;
--<var_name>;
<var_name>++;
<var_name>--;
```

Notation :

1). 前缀模式与后缀模式的返回值是不同的. 前缀模式返回变量进行了增量运算后的值, 后缀模式返回进行增量运算前的值. 如:

```
int i = 5, j = 5;
printf("%d, %d", ++i, j++);
```

该程序的返回值为:

```
6, 5
```

2). 自增、自减运算不能用于字面量

第二章　数组与字符串

§2.1　数组的定义及数组元素的引用

1. 数组的定义

```
<element_type> <array_name>[<array_length>];
```

Notation：

1). 数组索引从 0 开始

2). 通过 `sizeof(<array_name>)` 来获取数组的长度

2.1.1　二维和多维数组

1. 数组存储规则对于多维数组, 其存储规则可以概括为: 序号越靠右, 存储优先级越高. 此外, 若将数组的序号看成是一个多进制数, (e.g., `[8][9][2]` 看成数字 892), 并记该数为 a, 则存储顺序是按照 a 增加的顺序来的.