第3章 数据类型与表达式

如何让计算机下棋？为了让计算机能够下棋，在机器博弈程序设计中，首要任务就是通过恰当的数据结构使棋类要素数字化，以亚马逊棋（界面如图 3-1 所示）为例，通常需要考虑棋盘大小，棋子坐标，棋局状态（有无走子、落子），棋规（着法规则、胜负规则）等情况，要表示这些信息，需要用到数据类型、常量、变量等相关知识。

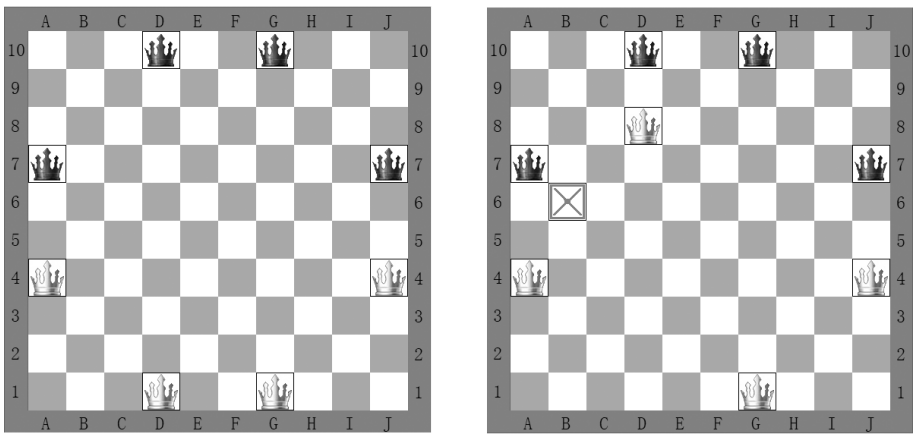


图 3-1　亚马逊棋博弈界面（白棋 D1 走到 D8，在 B6 放一个障碍）

在 C 语言中，所有数据都有自己的类型，数据类型是按被说明量的性质、表示形式、占据存储空间的多少、构造特点划分的。数据类型可分为基本数据类型、构造数据类型、指针类型和空类型四大类，具体如表 3-1 所示。

表 3-1　C 语言的数据类型

序号	类　型	描　述
1	基本数据类型	基本数据类型是自我说明的，即其值不可以再分解为其他类型，包括整型、字符型、实型（浮点型）、枚举类型

续表

序号	类　　型	描　　述
2	构造数据类型	亦称为聚合类型，是根据已定义的一个或多个数据类型用构造的方法定义的。即一个构造类型的值可以分解成若干个“成员”或“元素”。每个“成员”都是一个基本数据类型或是一个构造类型。它包括数组类型、结构类型、共用体(联合)类型
3	指针类型	指针是一种特殊的，同时又具有重要作用的数据类型。其值用来表示某个变量在内存中的地址。虽然指针变量的取值类似于整型量，但这是两个类型完全不同的量，因此不能混为一谈
4	空类型	用 void 说明，表明没有可用的值。它通常用于以下几种情况：函数返回值为空、函数参数为空、指针指向为空

本章接下来的内容结合机器博弈的应用，先介绍基本数据类型中的整型、浮点型和字符型，其他几种数据类型会在后面章节中进行讲解。


扫一扫

3.1 棋局要素

在机器博弈程序开发过程中，通常要重点考虑如下几部分内容：棋局博弈状态的表示、博弈平台或界面、着法搜索算法与局面评估函数。

其中，棋局博弈状态的数据表示直接影响程序的时间及空间复杂度，为了追求更高的效率，针对不同棋类的特点，可采用多种不同的表示方法。例如，棋盘可以考虑用数组(将在第6章介绍)表示，也可以用位棋盘(bit boards，将在第13章介绍)表示。博弈平台或界面最基本的功能是显示博弈过程，界面设计是否合理，将直接影响人机交互的流畅性。此外，还要判断博弈双方的行棋是否符合博弈项目的规则，以及最终胜负结果。着法搜索算法与局面评估函数是机器博弈系统开发中体现程序 AI 水平的关键技术，着法搜索算法的功能是从指定范围内的合法着法中找出最优解，局面评估函数用于评价局面优劣，二者通常配合使用。

在机器博弈开局时，博弈双方局面状态通常采用某种定式，例如中国象棋或亚马逊棋棋盘的初始布局，通常可以用常量表示此类开局状态。

但在机器博弈程序运行过程中，棋局状态不断发生改变，例如棋子的位置、状态(吃子、走子等)，对应的信息显然不能再用常量表示，此时博弈局面数字化表示需要用到变量。

以图 3-1 所示的亚马逊棋博弈界面为例，可以用一个 10×10 的二维数组 chessboard[10][10]表示棋盘，数组中每一个元素表示棋盘上一个格子的状态。0 表示该格子里没有棋子，1 表示该格子里有一个黑棋，2 表示该格子里有一个白棋，3 表示该格子里有一个障碍。棋盘初始化语句如下。

```
int  chessboard[10][10] ={{0,0,0,1,0,0,1,0,0,0},{0,0,0,0,0,0,0,0,0,0},
                          {0,0,0,0,0,0,0,0,0,0},{1,0,0,0,0,0,0,0,0,1},
                          {0,0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0,0,0},
                          {2,0,0,0,0,0,0,0,0,2},{0,0,0,0,0,0,0,0,0,0},
                          {0,0,0,0,0,0,0,0,0,0},{0,0,0,2,0,0,2,0,0,0}};
```

白棋 D1 走到 D8，在 B6 放一个障碍后，棋局状态数据如下：

```
{{0,0,0,1,0,0,1,0,0,0},{0,0,0,0,0,0,0,0,0,0},
 {0,0,2,0,0,0,0,0,0,0},{1,0,0,0,0,0,0,0,0,1},
 {0,3,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0,0,0},
 {2,0,0,0,0,0,0,0,0,2},{0,0,0,0,0,0,0,0,0,0},
 {0,0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,2,0,0,0}}
```

3.2 常量与变量

对于基本类型的数据，根据其值在程序运行过程中是否允许被改变，可分为常量和变量。在程序执行过程中，其值不允许发生改变的量称为常量，而其值可变的量则称为变量。在C程序中，变量必须先定义后使用。可以把数据类型与常量、变量相结合进行分类，例如，整型常量、整型变量，浮点型常量、浮点型变量。

3.2.1 常量

常量可以是整型常量、浮点型常量、字符型常量，以及枚举常量等任何的基本数据类型。例如：

(1) 整型常量，如123、0、－123。

(2) 浮点型常量，如1.23、0.0、－1.23。

(3) 字符型常量，如'C'、'G'、'0'。

1. 整型常量

在C语言中，整型数据在计算机的内存中是以补码形式存放的。通常使用的整数有十进制数、八进制数和十六进制数3种，在程序中根据前缀区分各种进制数。

1) 十进制数

十进制整数没有前缀，其数码为0～9。例如，1234、－1234、0是合法的十进制整数。

2) 八进制数

八进制整数通常是无符号数，且以0作为前缀，数码取值为0～7。例如，016(十进制数为14)、0101(十进制数为65)是合法的八进制数。

3) 十六进制数

十六进制整数的前缀为0X或0x，其数码取值为0～9、A～F或a～f。例如，0X12(十进制数为18)、0XFFFF(十进制数为65535)是合法的十六进制整数。

无符号数也可用后缀表示，整型常数的无符号数的后缀为“U”或“u”。例如：123u、0x12Au、123Lu均为无符号数。前缀、后缀可同时使用，以表示各种类型的数，如0X123Lu表示十六进制无符号长整数123。

2. 浮点型常量

浮点型也称为实型。浮点型常数也称为实数或者浮点数，由整数部分、小数点、小数部分和指数部分组成。在C语言中，实数只采用十进制。它有两种形式：小数形式和指数形式。

1) 小数形式

当使用小数形式表示时，由数码0～9和小数点组成，且至少包含整数部分、小数部分两

者之一。例如，0.0、0.123、1.23、123.0、123.、－1.23 等均为合法的实数。

2）指数形式

当使用指数形式表示时，由十进制数加阶码标志“E”或“e”以及阶码（只能为整数，可以带符号）组成，且至少包含小数点、指数两者之一。例如，实数正确的书写形式如下。

1.23E4（等于 1.23×10^4）、1.23E-4（等于 1.23×10^{-4}）、－0.123E4（等于 -0.123×10^4）。

以下是不合法的实数：

123（无小数点）、E123（阶码标志 E 之前无数字）、123.－E4（负号位置不对）、1.23E（无阶码）。

标准 C 允许使用后缀“f”或“F”表示该数为浮点数，如 123f 和 123.等价。

【注意】 浮点型常数都按双精度 double 型处理。

3. 字符型常量

字符型常量是用单引号括起来的一个字符。例如，'a'、'1'、'＋'、'！ '，都是合法的字符数据。字符型常量具有如下特点。

（1）字符型常量只能用单引号括起来，不能用其他符号。

（2）字符型常量只能是单个字符，不能是字符串。

（3）字符型常量可以是字符集中的任意字符。但数字被定义为字符型之后就不能参与数值运算。

【注意】 '1'与 1 不同，'1'是一个字符。

在 C 语言中，有一种特殊的字符，称为转义字符。它以反斜线“\”开头，后跟一个或几个字符，含义不同于原有的字符。转义字符主要用来表示那些用一般字符不便于表示的控制代码。C 语言常用的转义字符及其含义如表 3-2 所示。

表 3-2 C 语言常用的转义字符及其含义

转义字符	转义字符的意义	ASCII 代码
\n	回车换行	10
\t	横向跳到下一制表位置	9
\b	退格	8
\r	回车	13
\f	走纸换页	12
\\	反斜线符“\”	92
\'	单引号符	39
\"	双引号符	34
\a	鸣铃	7
\ddd	1～3 位八进制数所代表的字符	
\xhh	1～2 位十六进制数所代表的字符	

广义地讲，C 语言字符集中的任何一个字符均可用转义字符表示。表中，\ddd 和\xhh 中，ddd 和 hh 分别为八进制和十六进制的 ASCII 码值，如\101 表示字母'A'，\134 表示反斜

线,\XOA 表示换行等。

4. 字符串常量

字符串常量是由一对双引号括起的字符序列。例如,"Computer Games"和"勤劳的中华儿女"等都是合法的字符串常量。字符串常量和字符常量是不同的量。它们之间主要有以下区别。

(1) 字符常量由单引号括起来,字符串常量由双引号括起来。

(2) 字符常量只能是单个字符,字符串常量则可以含一个或多个字符。

(3) 可以把一个字符常量赋予一个字符变量,但不能把一个字符串常量赋予一个字符变量。在 C 语言中没有相应的字符串变量。这是与 Basic 语言不同的。但是可以用一个字符数组存放一个字符串常量,此内容将在第 6 章予以介绍。

(4) 字符常量占一字节的内存空间。字符串常量占的内存字节数等于字符串中的字节数加 1。增加的一字节中存放字符'\0'(ASCII 码值为 0),这是字符串结束的标志。例如:

字符串 "Computer Games" 在内存中所占的字节为

C	o	m	p	u	t	e	r		G	a	m	e	s	**\0**

字符常量'a'和字符串常量"a"虽然都只有一个字符,但在内存中的情况是不同的。'a'在内存中占一字节,可表示为

a

"a"在内存中占两字节,可表示为

a	\0

5. 符号常量与常变量

在 C 语言中,可以用一个标识符表示一个常量,通常称之为符号常量。符号常量就像常规的变量,只不过符号常量的值在定义后不能进行修改,且符号常量在使用之前必须先定义,其一般形式如下:

```
#define  标识符  常量
```

例如:

```
# define PI 3.1415926
```

其中,#define 是一条预处理命令(预处理命令都以“#”开头),称为宏定义命令(在第 8 章进一步介绍),其功能是把该标识符定义为其后的常量值。符号常量与变量不同,它的值在其作用域内不能改变,也不能再被赋值。一经定义,在以后的程序中,所有出现该标识符的地方均以该常量值代之。显然,使用符号常量的好处是:含义清楚,且可以“一改全改”。

除了使用#define 预处理命令,还可以使用 const 关键字定义常量。

使用 const 前缀声明指定类型的常量,其一般形式如下:

```
const 数据类型变量名=值;
```

例如：

```
const int LENGTH=10;      //定义棋盘长度
const int WIDTH=10;       //定义棋盘宽度
const char status='Y';    //定义棋局状态
```

【说明】

用 const 定义常量，实际上是改变变量的存储状态，是其值不允许改变的常变量，在编译、运行阶段会执行类型检查。

用 #define 定义的常量，是不带类型的常数，仅进行简单的字符替换，在预编译阶段起作用，不做任何类型检查。

3.2.2 变量

程序运行期间，其值可以改变的量称为变量，它其实是程序可操作存储区的名称。一个变量有一个名字，在内存中占据一定的存储单元。在变量使用之前必须对变量定义，一般放在函数体的开头部分。

C 语言中，每个变量都有特定的类型，类型决定了变量存储的大小和布局，该范围内的值都可以存储在内存中，运算符可应用于变量上。

变量的名称可以由字母、数字和下画线字符组成，必须以字母或下画线开头。因为 C 语言对大小写敏感，所以大写字母和小写字母是不同的。基于前面讲解的基本类型常量，有如下几种基本的变量类型。

1. 整型变量

(1) 基本型：类型说明符为 int。

(2) 短整型：类型说明符为 short int 或 short。

(3) 长整型：类型说明符为 long int 或 long。

(4) 无符号型：类型说明符为 unsigned。无符号型又可与上述 3 种类型匹配而构成。

- 无符号基本型：类型说明符为 unsigned int 或 unsigned。
- 无符号短整型：类型说明符为 unsigned short。
- 无符号长整型：类型说明符为 unsigned long。

各种无符号类型量所占的内存空间字节数与相应的有符号类型量相同。但由于省去了符号位，故不能表示负数。整型变量的存储大小和取值范围如表 3-3 所示。

表 3-3 整型变量的存储大小和取值范围(VS 2010 环境)

类型说明符	存储大小	值 范 围
int	4B	−2 147 483 648 到 2 47 483 647，即 $-2^{31}\sim(2^{31}-1)$
unsigned int	4B	0 到 4 294 967 295，即 $0\sim(2^{32}-1)$
short	2B	−32 768 到 32 767，即 $-2^{15}\sim(2^{15}-1)$
unsigned short	2B	0 到 65 535，即 $0\sim(2^{16}-1)$
long	4B	−2 147 483 648 到 2 147 483 647，即 $-2^{31}\sim(2^{31}-1)$
unsigned long	4B	0 到 4 294 967 295，即 $0\sim(2^{32}-1)$

【注意】 各种类型的存储大小与系统位数有关，为了得到某个类型或某个变量在特定平台上的准确大小，可以使用 sizeof 运算符的表达式 sizeof(变量或类型)得到变量或类型存储字节的大小。

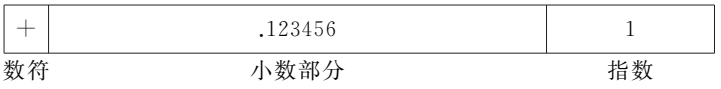
2. 浮点型变量

浮点型变量分为单精度(float 型)、双精度(double 型)和长双精度(long double 型)3类，其所占内存空间及数值范围如表 3-4 所示。

表 3-4 浮点型数据占用空间及数值范围(VS 2010 环境)

类型说明符	存储大小	值 范 围
float	4B	$1.2\times10^{-38}\sim 3.4\times10^{38}$
double	8B	$2.3\times10^{-308}\sim 1.7\times10^{308}$
long double	8B	$2.3\times10^{-308}\sim 1.7\times10^{308}$

浮点型数据在内存中一般按指数形式存储。例如，浮点数 1.23456 在内存中的存放形式如下：



- 小数部分占的位数越多，数的有效数字越多，精度越高。
- 指数部分占的位数越多，则能表示的数值范围越大。

3. 字符型变量

字符型变量用来存储单个字符，其类型说明符是 char。每个字符型变量被分配一字节的内存空间，只能存放一个字符。字符型变量所占存储空间及取值范围如表 3-5 所示。

表 3-5 字符型变量所占存储空间及取值范围

类型说明符	存储大小	值 范 围
char	1B	−128～127 或 0～255
unsigned char	1B	0～255
signed char	1B	−128～127

字符以 ASCII 码的形式存放在变量的内存单元中，可以把它们看成整型量。C 语言允许字符变量参与数值运算，即用字符的 ASCII 码值参与运算。可以对整型变量赋以字符值，也可以对字符变量赋以整型值。在输出时，允许把字符变量按整型量输出，也允许把整型量按字符量输出。字符量为单字节量，当整型量按字符型量处理时，只有低字节参与处理。

3.2.3 变量的定义与声明

1. 变量定义

变量定义就是告诉编译器在何处创建变量的存储，以及如何创建变量的存储。变量定

义指定一个数据类型，并包含了该类型的一个或多个变量的列表。其一般形式如下：

```
类型说明符 变量名标识符 1, 变量名标识符 2, …, 变量名标识符 n;
```

在这里，类型说明符必须是一个有效的 C 语言数据类型，可以是 char、int、float、double 或任何用户自定义的对象，**变量名标识符**可以由一个或多个标识符名称组成，多个标识符之间用逗号分隔，且类型说明符与变量名之间至少用一个空格间隔。下面列出几个有效的定义。

```
int x,y;                    //x、y 为整型变量，可以表示棋子走子坐标
float power;                //power 为单精度实型变量，可以表示局面评估值
double m,n;                 //m、n 为双精度实型变量
char a,b;                   //a、b 为字符型变量
```

【注意】 由于实型变量是由有限的存储单元组成的，能提供的有效数字总是有限的，因此，在运算过程中，实型数据可能存在舍入误差。例如，1.0/3 * 3 的结果并不等于 1。

2. 变量声明

变量声明向编译器保证变量以指定的类型和名称存在，这样，编译器在不需要知道变量完整细节的情况下也能继续进一步编译。变量声明只在编译时有意义，在程序连接时编译器需要实际的变量定义，即变量声明之后，需要定义才能使用。

变量的声明有两种情况：

(1) 一种是需要建立存储空间的。例如，int x,y; 在声明的同时就已定义，并分配了存储空间。

(2) 另一种是不需要建立存储空间的，通过使用 extern 关键字声明变量名，而不定义它。例如：

```
extern int x; //声明一个全局变量 x，变量 x 可在其他文件中定义
```

变量定义与声明的区别在于：

变量定义也是声明，变量定义为变量分配存储空间，而声明不会。变量定义还可为变量指定初始值。变量声明用于向程序表明变量的类型和名字。程序中变量可以声明多次，但只能定义一次。

3.2.4 变量初始化

在 C 语言程序中常常需要对变量赋值，以便使用变量。变量可以在定义的时候被初始化(指定一个初始值)。变量初始化由赋值符号后跟一个常量表达式组成，其一般形式如下：

```
类型说明符 变量 1=值 1,变量 2=值 2,……;
```

下面列举几个定义并初始化的实例：

```
int x=1,y=2;
float a=3.2,b=3f;
char str1='C',str2='G';
```

【注意】 在定义中不允许连续赋值，如 int a=b=c=5 是错误的。

对于未初始化的变量定义，带有静态存储持续空间的变量默认被初始化为 NULL(值

为 0),其他变量的初始值都是不确定的。

例 3.1 2020 年 11 月 24 日,总质量 8.2 吨的“嫦娥五号”探测器,由“长征五号”运载火箭成功发射。这句话中的数字可以用什么基本数据类型变量存储?

【分析】 年月日可以用整型变量存储,探测器总质量可以用浮点型变量存储。

各变量可以定义如下:

```
int year=2020,month=11,date=24;
float weight=8.2;
```

3.3 数据类型转换

求解一个表达式值时,如果一个表达式中含有不同类型的数据,最终会转换为同种类型数据。数据类型转换的方法有两种:一是隐式转换,由编译器自动执行;二是显式转换,通过使用**强制类型转换运算符**实现。编程时,在需要类型转换时使用强制类型转换运算符,是一种良好的编程习惯。

3.3.1 隐式类型转换

隐式类型转换由编译系统自动完成,不同类型的数据混合运算时,遵循以下转换规则,如图 3-2 所示。

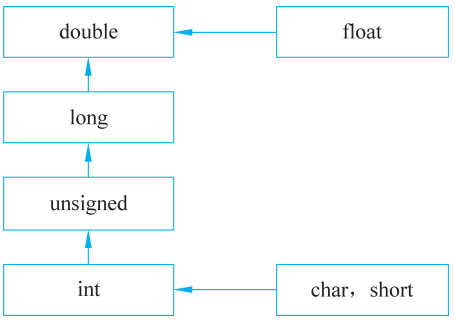


图 3-2 隐式类型转换规则图

(1) 如果一个运算符两边的数据类型不同,先按数据长度增加的方向将其转换成同一数据类型,再进行运算,以保证不降低精度。如 char 型和 short 型参与运算时,必须先转换成 int 型。int 型和 long 型运算时,先把 int 型量转成 long 型后再进行运算。

(2) 浮点运算都以双精度进行,即使仅含单精度量运算的表达式,也要先转换成双精度型,再进行运算。

(3) 在赋值运算中,如果赋值运算符两边数据类型不相同,系统将自动进行类型转换,即把赋值符号右边量的类型转换为左边量的类型。

① 如果右边量的数据类型长度比左边长时,将丢失一部分数据,这样会降低精度。

② 整型量赋给浮点型变量,数值不变,但将以浮点形式存储,即增加小数部分(小数部分的值为 0)。

③ 字符型量赋给整型变量，仅将字符的 ASCII 码值放到整型变量低八位中，高位为 0。

④ 整型量赋给字符型变量，只把低八位赋予字符变量。

例如，有如下语句：

```
int s,r=5;
s=r*r*3.14;
```

其中，变量 s、r 为整型。在执行 s=r*r*3.14;语句时，r 转换成 double 型计算，结果也为 double 型。但由于 s 为整型，故赋值结果仍为整型，舍去了小数部分。

例 3.2 给定华氏温度为 30，计算并输出相应的摄氏温度(浮点型数据)。已知华氏温度 F 与摄氏温度 C 间的计算公式为 $C=\frac{5}{9}(F-32)$。

【分析】 定义华氏温度为整型变量，相应的摄氏温度为浮点型数据，计算过程中数据类型自动转换为浮点型。参考程序如下：

```
01  #include<stdio.h>
02  int main(void)
03  {
04      int f=30;                          //定义整型变量 f 存储华氏温度
05      float c=5.0/9*(f-32);              //定义浮点型变量 c 存储摄氏温度
06      printf("摄氏温度=%f\n",c);         //输出浮点型数据
07      return 0;
08  }
```

【思考】 将程序第 5 行的 5.0 换为 5，结果会如何？

3.3.2 显式类型转换

显式类型转换，即强制类型转换，把表达式显式地从一种类型强制转换为另一种数据类型，一般形式如下：

```
(类型说明符)(表达式)
```

例如，把一个 float 型数据 a 转换为整型，可以用(int)a 显式类型转换。

使用强制转换时应注意以下问题：

(1) 类型说明符和表达式都必须加括号(单个变量可以不加括号)，如把(int)(a+b)写成(int)a+b 则成了把 a 转换成 int 型之后再与 b 相加了。

(2) 无论是显式类型转换还是隐式类型转换，都只是为了本次运算的需要而对变量的数据长度进行的临时性转换，而不改变数据说明时对该变量定义的类型。

例如：

```
float pi=3.1415926;
```

(int)pi 的值强制转换为整数 3(舍去了小数部分)，但该值是临时性的，仅在运算中起作用，而 pi 本身的类型并不改变，其值仍为 3.1415926。

3.4 运算符和表达式

C 语言中运算符丰富，可以分为 10 类，包括算术运算符、关系运算符、逻辑运算符、位操作运算符、赋值运算符、条件运算符、逗号运算符、指针运算符、求字节数运算符、特殊运算符，具体如附表 A-1 所示。C 语言运算符有优先级和结合性两个属性。

1. 运算符的优先级

C 语言中，运算符的运算优先级由高到低共分为 15 级。在表达式中，优先级较高的先于优先级较低的进行运算，优先级相同时，则按运算符的结合性所规定的结合方向处理。

一般而言，单目运算符优先级较高，赋值运算符优先级较低。算术运算符优先级较高，关系和逻辑运算符优先级较低。

2. 运算符的结合性

运算符的结合性是指同一优先级的运算符在表达式中操作的组织方向，即当一个运算对象两侧运算符优先级相同时，运算对象与运算符的结合顺序，C 语言运算符的结合方向分为两种，即左结合性（自左至右的结合方向）和右结合性（自右至左的结合方向）。

大多数运算符的结合方向是从左至右，仅有三类运算符的结合方向是从右至左，即单目运算符（＋＋、－－、－、!）、条件运算符（?）以及赋值运算符（＝）。例如，x＝y＝z，由于“＝”的右结合性，应先执行 y＝z 再执行 x＝（y＝z）运算。C 语言运算符的优先级和结合性具体如附表 A-1 所示。

表达式是由常量、变量、函数和运算符组合起来的式子，单个常量、变量、函数可以看作表达式的特例。各运算量参与运算的先后顺序不仅要遵守运算符优先级别的规定，还要受运算符结合性的制约，以便确定是自左向右进行运算，还是自右向左进行运算。这种结合性是其他高级语言的运算符所没有的，因此也增加了 C 语言的复杂性。

3.4.1 算术运算符及其表达式

算术运算符包括“＋”“－”“＊”“/”“％”“＋＋”和“－－”。算术表达式则是用算术运算符和括号将运算对象（也称操作数）连接起来的、符合 C 语言语法规则的式子。在算术运算符中：

减法运算符“－”，为双目运算符时，具有左结合性；作负值运算符时，为单目运算，具有右结合性，如－x、－5 等。

除法运算符“/”，参与运算量均为整型时，结果也为整型，舍去小数。运算量中若有一个为实型，则结果为双精度实型。

求余运算符（模运算符）“％”要求参与运算的量均为整型。求余运算的结果等于两数相除后的余数。

自增、自减运算符“＋＋”和“－－”的功能是使变量值自增 1 或自减 1，它们均为单目运算符，都具有右结合性。自增、自减运算符形式如表 3-6 所示。

表 3-6　自增、自减运算符形式

运算式	含　义	运算式	含　义
++i	i 先自增 1 后，再参与其他运算	i++	i 先参与其他运算，i 的值再自增 1
--i	i 先自减 1 后，再参与其他运算	i--	i 先参与其他运算，i 的值再自减 1

假设 i 的当前值为 5，则 y=i++；语句运行后，y 值为 5，i 值为 6。若 i 的当前值为 5，则 y= ++ i；语句运行后，y 值为 6，i 值也为 6。

3.4.2　赋值运算符及其表达式

1. 简单赋值运算符和表达式

简单赋值运算符记为“=”，赋值运算符具有右结合性。由赋值运算符连接的式子称为赋值表达式。简单赋值表达式的功能是计算表达式的值后再赋予左边的变量。其一般形式如下：

```
变量=表达式
```

凡是表达式可以出现的地方均可出现赋值表达式。例如，x=y=c=(a=1)+(b=2)是合法的，可理解为 x=(y=(c=((a=1)+ (b=2))))。其意义是把 1 赋予 a，2 赋予 b；再把 a、b 相加的和赋予 c，故 c 值为 3；最后把 c 赋予 y，y 赋予 x。

在赋值表达式末尾加上分号，就构成赋值语句。

2. 复合赋值运算符

在赋值运算符“=”之前加上其他二目运算符可构成复合赋值运算符，如+=、-=、*=、/=、%=、<<=、>>=、&=、^=、|=。复合赋值表达式的一般形式如下：

```
变量 双目运算符=表达式
```

它等效于：

```
变量=变量 运算符 表达式
```

例如：

```
y*=x+1
```

等价于

```
y=y*(x+1)
```

复合赋值运算符的写法，有利于编译处理，能提高编译效率，并产生质量较高的目标代码。

例 3.3　语句 y=(x=a+b),(b+c),(a+c)；运行后，求表达式的结果。

【分析】　在 C 语言中，逗号“,”也是一种运算符，称为逗号运算符。其功能是把前后两个表达式连接起来，组成一个表达式，该表达式称为逗号表达式，其一般形式如下：

```
表达式 1,表达式 2,…,表达式 n
```

其求值过程是分别求各个表达式的值，并以最后一个表达式 n 的值作为整个逗号表达式的值。程序中使用逗号表达式，通常要分别求逗号表达式内各个表达式的值，但有时候并不一定要求整个逗号表达式的值。本例中，y 等于整个逗号表达式的值，也就是表达式(a+c)的值。

【注意】 并不是在所有出现逗号的地方都组成逗号表达式，如在变量定义中、函数参数列表中，逗号只是用作各变量之间的间隔符。


扫一扫

3.4.3 机器博弈中的局面评估函数

机器博弈中，评价当前局面是否对己方有利，通常需要借助评估函数。评估函数通常可以用表达式形式描述，表达式中的各个变量分别代表不同的因素和权重系数。设计评估函数需要考虑诸多因素，在完备信息博弈中双方的子力(Material)、领地(Territory)、位置(Position)、空间(Space)、机动性(Mobility)、拍节(Tempo)、威胁(Threat)、形状(Shape)、图案(Motif)都可以作为评估参数，非完备信息博弈中除了己方已知参数外，还要猜测对手的情况，并通过量化后加权组合而成。

以亚马逊棋为例，根据规则，博弈双方目标是用自己的棋子和障碍将对手的棋子困住，因此一般采取占领地或阻塞对手路线的思路设计评估函数。亚马逊棋评估函数，一般基于 Kingmove 和 Queenmove 走法设计，下面给出了一种典型的评估函数，其表达式形式如下：

```
Value=a*t1+b*t2/2+c*((c1+c2)/2)+s
```

其中，t1、t2 为 Territory 特征值，代表双方对空闲区域的控制能力；c1、c2 为位置 Position 特征值，用于反映双方对空格控制权的差值特征；a、b、c 参数是根据棋局进行状态不断变换的权重，用于动态控制在不同棋局状态下各个特征值对结果的影响；参数 s 用于判断当前着法是否会产生区域浪费(某个空区域被围堵，不可使用)的情况。

尽管对博弈局面评估得越全面、越准确，获胜的概率就会越高，但在机器博弈中，有个很重要的约束条件——时间。在局面评估中考虑的情况越全面、越细致，则耗费的时间就越多，搜索的深度和速度必然受到影响。


扫一扫

3.5 输入与输出

顺序结构是结构化程序设计中最为简单的一种，语句按照位置的先后顺序执行。本节通过机器博弈中一些简单的顺序结构程序实例，介绍如何通过调用函数库中的字符输入、输出函数和格式输入、输出函数，实现棋局信息的输入和输出操作。

3.5.1 字符输入/输出函数

1. 字符输出函数 putchar

字符输出函数的一般形式如下：

```
int putchar(char c)
```

功能：向终端输出一个字符，并返回该字符的 ASCII 码值。

2. 字符输入函数 getchar

字符输入函数的一般形式如下：

```
int getchar( )
```

功能：接收从终端输入的一个字符，并返回该字符的 ASCII 码值。

3.5.2 棋局信息输出

例 3.4 在机器博弈比赛项目中，爱恩斯坦棋博弈通过掷骰子的方式决定对弈双方可移动的棋子。编写程序，随机产生一个骰子值，并在屏幕上输出该值。

【分析】 putchar 函数仅用于单个字符的输出，如果需要输出更多类型的数据，一般采用格式输出函数 printf。printf 函数按照格式控制要求，向终端输出参量表中各个输出项的值，其一般格式如下：

```
printf(格式控制字符串,参量表);
```

其中，“格式控制字符串”是用双引号括起来的字符串，包括两种信息：①格式控制说明，由'%'字符和格式字符组成；②需要原样输出的普通字符。参量表可以没有（输出一个字符串常量时），也可以包含一个或多个常量、变量或表达式，多个参量用逗号分隔。

另外，产生随机整数可以使用 rand 函数实现，随机数的范围应控制在 1～6；printf 函数可实现输出该数到屏幕。

```
01  #include<stdio.h>
02  #include<stdlib.h>
03  #include<time.h>
04  int main(void)
05  {
06      int magic;
07      srand((unsigned int)time(NULL));          //初始化随机数发生器
08      magic =rand() %6 +1;                      //产生一个 1~6 的随机数
09      printf("random  number =%d\n", magic);    //输出随机数到屏幕
10      return 0;
11  }
```

程序运行结果如下：

```
random  number=4
```

【说明】

（1）rand 函数用于产生随机整数。为了避免产生重复的有序随机数，需要使用 srand(seed)函数初始化随机数发生器，用时间作种子 srand(time(NULL))，可以保证随机数的随机性。由于使用了时间函数，因此需要引入头文件 time.h。

（2）在标准 C 中，rand 函数产生随机数的范围是 0～RAND_MAX(RAND_MAX 值为 32767)，而符号常量 RAND_MAX 是在头文件 stdlib.h 中定义的，因此，使用随机函数需要为程序增加头文件 stdlib.h。

（3）如果要改变产生随机数的范围，可以通过一些简单计算来实现。如产生一个在区间[a,a+b−1]的随机数，可以用表达式 rand( )%b+a 实现。

例如，上述程序中产生一个 1～6 的随机数，可表示为

```
magic=rand( ) %6 +1;
```

在 printf 语句中，"magic = %d\n"格式控制说明“%d”将参量“magic”转换成十进制整数的形式输出，“random number =”是普通字符，会原样输出到屏幕。其他更多的格式控制符说明如表 3-7 所示。

表 3-7 格式控制符说明

格式控制符	功 能 说 明
%d	输入或输出带符号的十进制整数，正数的符号省略
%u	输入或输出无符号的十进制整数
%o	输入或输出无符号的八进制整数，不输出前导符 0
%x	输入或输出无符号的十六进制整数(字母小写)，不输出前导符 0x
%X	输入或输出无符号的十六进制整数(字母大写)，不输出前导符 0X
%c	输入或输出一个字符
%s	输入或输出字符串。输入时以非空白字符开始，以第一个空白字符结束，空白字符指空格、回车和制表符
%f	以十进制小数形式输出单、双精度实数，默认输出 6 位小数，整数部分按实际位数输出；用于输入时，输入小数或指数形式均可
%e，%E	以指数形式输出实数，用 E 时，指数用大写字母表示；用于输入时可与 f、g 互相替换，大小写作用相同
%g，%G	选取%f 或%e 中输出宽度较小的一种使用，不输出无意义的 0，用 G 时，若以指数形式输出，指数用大写字母表示；用于输入时可与 e、f 互相替换，大小写作用相同

例 3.5 五子棋棋盘如图 3-3 所示，输出符合棋谱规范的一步棋谱 B(H,8)。

```
01  #include<stdio.h>
02  int main(void)
03  {
04      int y;
05      char x,color;
06      color='B';
07      x='H';
08      y=8;
09      printf("行棋棋谱:");
10      printf("%c(%c,%d)",color,x,y);   //输出一步棋谱信息
11      return 0;
12  }
```

程序运行结果如下：

```
行棋棋谱：B(H,8)
```

【说明】

(1) printf 函数既可以输出字符串常量，也可以使用格式控制符进行格式化输出。程序

第 10 行 printf 函数中的 3 个格式符“%c”“%c”“%d”与参量表中的 3 个变量按先后顺序一一对应，分别控制 3 个变量的输出格式。

（2）在机器博弈竞赛中，保存棋谱有利于比赛过程回溯与算法改进。2018 年，中国人工智能学会机器博弈专业委员会制定了包括五子棋、六子棋、爱恩斯坦棋等十几个棋（牌）类项目的棋谱规范。本例中并未完全参照棋谱规范输出完整的棋谱信息，而是进行了信息截取和适当调整。

（3）在 printf 函数的格式说明中，可以在%和格式符中间插入几种格式修饰符，用来对输出格式进行微调，如表 3-8 所示。

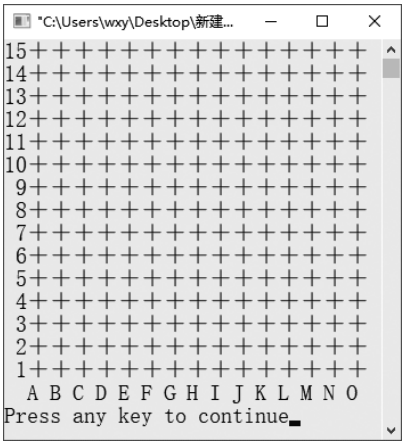


图 3-3　五子棋棋盘

表 3-8　printf 函数格式修饰符

格式修饰符	说　明
英文字母 i	加在格式符 d、u、o、x 之前，用于输出 long 型数据
输出域宽 m	m 表示一个整数，指定输出项输出时所占的列数（包括符号位） 若 m 为正整数，当数据的实际位数小于 m 时，左端补空格；当数据的实际位数大于 m 时，按实际位数输出；若 m 有前导符 0，则左边多余位补 0 若 m 为负整数，当数据的实际位数小于 m 的绝对值时，右端补空格
显示精度.n	n 表示一个大于或等于 0 的整数，精度修饰符一般用在域宽修饰符之后 对于浮点数，用于指定输出浮点数的小数位数 对于字符串，用于指定从字符串左侧开始截取的字符个数

例如，若执行下面的程序段：

```
int camera=4000;
float mscreen=6.53;
printf("%6d,%6.3f\n", camera,mscreen);
```

则运行后的输出结果是：

```
4000,6.530
```

上述语句中的“%6d”规定了变量 camera 的输出列数为 6 列，由于该变量的实际位数为 4，因此，输出时在数值的左侧补了两个空格符。语句中的“%6.3f”规定了变量 mscreen 输出 6 列，同时保留 3 位小数，而 6.530 实际为 5 列，因此，输出时也在左侧补了一个空格符。

【注意】　如果想输出字符'%'，应该在格式控制字符串中用两个连续的%表示，如：

```
printf("%f%%",45.8/60*100);
```

输出：

```
76.333333%
```

3.5.3　棋局信息输入

例 3.6　在五子棋人机博弈中，黑方（人方）输入一步行棋，棋盘样式参照图 3-3。

【分析】 棋局信息的输入可以采用格式输入函数 scanf，其一般格式如下：

```
scanf(格式控制字符串,地址列表);
```

功能：按照格式控制要求，将输入的数据赋值给地址列表中的各个变量。其中，“格式控制字符串”的含义与 printf 函数类似；“地址列表”是若干个变量地址构成的列表，地址之间用逗号分隔。

```
01  #include<stdio.h>
02  int main(void)
03  {
04      int y;
05      char x,color;
06      printf("请输入落子坐标,格式:color,X,Y\n ");
07      scanf("%c%c%d",& color,&x,&y);
08      printf("黑方行棋棋谱为:\n");
09      printf("%c(%c,%d)",color,x,y);
10      return 0;
11  }
```

程序运行结果如下：

```
请输入落子坐标,格式: color,X,Y
BH8↙
黑方行棋棋谱为:
B(H,8)
```

【说明】

（1）scanf 函数中，“&color”里面的“&”是“地址运算符”，& color 表示变量 color 在内存中的地址。该例中 scanf 函数的作用是：将输入的 3 个数据按照变量 color、X 和 Y 的内存地址存进去。

（2）scanf 函数中格式控制字符之间无间隔时，可用空格键、Enter 键、Tab 键对输入的数据进行间隔。若 scanf 函数格式控制字符串中包含其他的字符常量，如逗号、等号和字母等，则在输入数据时，需要在对应位置输入与这些字符相同的字符。但在用“%c”格式输入字符时，空格字符和转义字符都作为有效字符输入。

输入函数的格式控制符说明如表 3-7 所示。scanf 函数格式修饰符如表 3-9 所示。

表 3-9 scanf 函数格式修饰符

格式修饰符	说　明
英文字母 i	加在格式符 d、u、o、x 之前，用于输入 long 型数据；加在格式符 f、e 之前，用于输入 double 型数据
英文字母 h	加在格式符 d、o、x 之前，用于输入 short 型数据
输入域宽 m	m 表示一个正整数，指定输入数据所占的列数
*	表示对应的输入项输入后不赋给相应的变量

【说明】

（1）可以在输入时指定域宽，使系统自动截取所需的数据。例如：

```
scanf("%2d%3d",&x,&y);
输入:234579↙
```

则系统取前两列赋给变量 x,取第 3 至 5 列赋给变量 y,即 x 值为 23,y 值为 457。

(2) 输入数据时不允许指定精度,例如:

```
scanf("%5.2f",&t);
```

是不合法的。

(3) 如果在%后面有字符'*',表示跳过它指定的列数。例如:

```
scanf("%2d%*2d%3d",&x,&y);
输入:234579628↙
```

则系统将 23 赋给变量 x,然后跳过 2 列,再取 796 赋给变量 y。%*2d 表示读入 2 位整数,但不赋给任何变量。

3.6 小结

本章主要介绍了如下内容:

(1) C 语言基本数据类型的分类及特点。

(2) 机器博弈中的棋局要素,常量与变量的定义和初始化。

(3) 数据类型转换规则。

(4) 常见运算符及其表达式,机器博弈中的评估函数;表达式是由运算符连接常量、变量、函数所组成的式子,表达式也有值和类型。

(5) 基本输入/输出函数及棋局信息的输入与输出。

3.7 习题

1. 编写一个 C 语言程序,求下列表达式的值,并与手工计算结果相比较。

(1) 若有 int a=7; float x=2.5,y=4.7; 求 x+a%3*(int)(x+y)%2/4。

(2) 若有 int a=2,b=3; float x=3.5,y=2.5; 求 (float)(a+b)/2+(int)x%(int)y。

(3) 若有 int x=3,y=4,z=5; 求 (x&&y)= =(x||z)。

(4) 若有 int x=3,y=4,z=5; 求!(x>y)+(y! =z)||(x+y)&&(y-z)。

2. 参考图 3-1 所示亚马逊棋博弈界面,根据需要定义几个变量并初始化,用于存放一步行棋信息。(提示:棋局描述可以考虑棋子颜色、棋子坐标、障碍坐标等信息。)

3.8 扩展阅读——机器博弈竞赛

1. 机器博弈竞赛概述

机器博弈涉及知识库、搜索算法、局面评估、神经网络与机器学习等多种技术，吸引了国内外越来越多的专家、学者与机器博弈爱好者参与到相关技术的研究中。而机器博弈竞赛则为相关技术人员提供了一个公平、开放的交流与验证的平台。

目前，机器博弈竞赛涵盖多种类型的博弈项目：

（1）按参与人数划分，包括双人博弈（如中国象棋、围棋）和多人博弈（如二打一扑克）；

（2）按参与人对信息的掌握程度划分，包括完备信息博弈（如中国象棋、围棋、六子棋、亚马逊棋、苏拉卡尔塔棋等）和非完备信息博弈（如幻影围棋、军棋、德州扑克、桥牌、麻将等）；

（3）按参与人之间有无合作划分，包括合作博弈（如桥牌）与非合作博弈（如中国象棋）；

（4）按着法产生确定性划分，包括确定性博弈（如亚马逊）与随机性博弈（如爱恩斯坦棋、牌类游戏）。

2. 国际机器博弈竞赛

由国际机器博弈协会（International Computer Games Association，ICGA）组织的每年一届的国际计算机博弈比赛（Computer Olympiad，CO），至今已经有几十年的历史。比赛项目包括中国象棋、六子棋、亚马逊棋、围棋等。

除了以上竞赛，还有世界范围内的各种人机大战活动。其中最著名的比赛有国际象棋世界棋王加里·卡斯帕罗夫与 IBM 公司的计算机程序“深蓝”在 1996 年和 1997 年分别进行的两场国际象棋人机大战；2011 年，IBM 公司的超级计算机 Watson 在 Jeopardy 比赛中击败两位世界冠军；2016—2017 年，人工智能程序 AlphaGo 分别与韩国棋手李世石和中国棋手柯洁的围棋人机大战。人机大战，皆以计算机获胜为最终结局。

3. 中国机器博弈竞赛

2006 年 8 月，由中国人工智能学会主办，首届中国计算机博弈锦标赛拉开帷幕，这标志着中国机器博弈活动轰轰烈烈地展开，全速起航。从 2011 年开始，每年举行一届中国大学生计算机博弈大赛暨全国锦标赛，该赛事由中国人工智能学会和教育部高等学校计算机类专业教学指导委员会共同主办（大赛网址为 http://computergames.caai.cn/）。现在，“中国机器博弈竞赛”已经形成校赛、省赛、国赛三级体制，进入高教学会大学生机器人竞赛清单及其排行指数清单。中国机器博弈竞赛以“高技术、强对抗、有门槛、有热度”为特点，提升大学生创新能力，助力人工智能高技术型人才的培养。

目前，中国机器博弈竞赛共设置 19 个项目，分为仅面向大学生的机器博弈大赛项目和面向全社会的锦标赛项目两大类。其中，大学生竞赛项目包括五子棋、六子棋、点格棋、苏拉卡尔塔棋、亚马逊棋、幻影围棋、不围棋、爱恩斯坦棋、军棋、海克斯棋总计 10 种棋类；锦标赛项目包括：中国象棋、围棋（19 路）、国际跳棋（100 格）、国际跳棋（64 格）、二打一扑克牌（斗地主）、德州扑克、麻将、藏棋久棋和桥牌总计 9 种棋牌类。参赛队伍分别来自北京理工大

学、东南大学、东北大学、吉林大学、中央民族大学、北京邮电大学、北京科技大学、哈尔滨工程大学、安徽大学、西安电子科技大学、武汉理工大学、中国海洋大学、成都理工大学、沈阳航空航天大学等 100 多所高校及中科院相关研究院所。

4. 机器博弈研究与竞赛的意义

各种形式的机器博弈竞赛，激发了人们的挑战热情和创新精神，检验了新技术，促进了学术交流，为社会培养了大量的科技精英，极大地推动了机器博弈技术的研究。在促进人工智能快速发展的同时，还产生了新的科研成果。

特别值得一提的是，尽管我国机器博弈竞赛的发展历程相对短暂，但是发展迅猛。尤其是在近几年的全国比赛中，平均每年有数百支代表队近千人参赛，参赛人数规模不逊于目前国际上任何一项机器博弈比赛赛事。目前，我们在科学研究方面与国际先进水平尚有差距，整体技术水平还有待提高，但是这种差距正在不断地缩小。在全球人工智能热潮大背景下，恰逢国家新一代人工智能发展规划发布的契机，面向全国大、中、小学生推广与开展机器博弈竞赛、宣传与科普人工智能知识，更具有时代意义。

相信不久的将来，在国内机器博弈领域的专家、学者与爱好者共同努力下，我国不仅将成为机器博弈活动的大国，而且会成为机器博弈技术的强国。