

第1章 C++的初步知识

- *1.1 C 和 C++语言属于计算机高级语言，支持面向过程的程序设计方法。
- *1.2 C++兼容 C，是 C 的加强版，增加了面向对象(OOP) 机制。
- *1.3 用高级语言编写的程序称作源程序，C++源程序的扩展名为 CPP，~~C 源程序的扩展名为 C。~~
- *1.4 源程序被编译或解释为机器语言方可执行。C 和 C++均采用编译方式，源程序经编译、连接后形成机器语言的目标程序文件。
- *1.5 用户最终执行的是目标程序文件，它是二进制可执行文件，扩展名为 EXE。
- *1.6 执行目标程序与源程序无关，但是，要修改目标程序，必须先修改源程序，然后，重新编译、连接，产生新的目标程序或覆盖原目标程序。
- 1.7 C 和 C++均支持模块化程序设计，C 以函数为基本模块，C++增加了类模块。
- *1.8 C 和 C++均支持结构化程序设计，有三种基本结构：顺序、分支、循环。
- *1.9 每个 C 或 C++源程序可由若干个文件组成，程序总是从主函数 main 开始执行，因此，每个源程序有且仅有一个主函数。
- *1.10 C 和 C++区分字母的大小写。
- *1.11 C 和 C++语句以分号结尾，仅有分号的语句称作空语句。
- *1.12 表达式后缀分号称作表达式语句。
- *1.13 用花括号 {} 括起来的一组语句统称复合语句。复合语句可省略其后分号，但复合语句内的语句不可省略分号(除非它也是复合语句)。
- *1.14 C 的注释以 /* 开头，结束于 */，可跨行。C++增加了以 // 开头，至行尾结束的注释。
- *1.15 C 和 C++能直接对内存进行操作，从而实现对硬件的操作。
- 1.16 广义地说，求解问题的方法和步骤称作算法。就程序设计而言，对数据的操作方法和步骤称作算法。
- *1.17 算法和数据结构是程序的两大要素。
- 1.18 掌握三种基本结构的图示。

第2章 数据类型和表达式

*2.1 掌握基本类型的类型名、缩写、存储量、取值范围。

类型名	类型标识符	存储量	取值范围
有符号短整数	signed short int	2 字节	-32768~32767
无符号短整数	unsigned short int	2 字节	0~65535
有符号长整数	signed long int	4 字节	-2147483648~2147483647
无符号长整数	unsigned long int	4 字节	0~4294967295
有符号字符	signed char	1 字节	-128~127
无符号字符	unsigned char	1 字节	0~255
单精度	float	4 字节	$\pm(3.4\text{E}-38\sim 3.4\text{E}38)$
双精度	double	8 字节	$\pm(1.7\text{E}-308\sim 1.7\text{E}308)$
长双精度	long double	8 字节	$\pm(1.7\text{E}-308\sim 1.7\text{E}308)$
逻辑值	bool	1 字节	0~1

- *2.2 由正号、负号、数字以及合法前缀和后缀组成的有效数称作整数(小含小数点、E、e 的数)。
- *2.3 含有小数点或 E 或 e 的数为双精度实型。
- *2.4 整型、实型、字符型、布尔型统称数值型。
- *2.5 整数后缀 U 或 u 为无符号整型。整数后缀 L 或 l 为长整型，实数后缀 L 或 l 为长双精度。
在 VC 6.0 中，整数默认为长整数，长双实数归入双精度，因此，整数和实数的后缀 L 均可省略。
- *2.6 0X 或 0x 打头的整数为十六进制整数。0 打头的整数为八进制整数。
- *2.7 浮点数“尾数 E 阶码”表示“尾数 $\times 10^{\text{阶码}}$ ”，其中，阶码必须是整数。
- *2.8 用一对单引号括起来的单个普通字符或转义字符称作字符常量，以其 ASCII 码存储。记住数字和字母的 ASCII 码。
字符是 1 字节整数，但默认输出形式是字符。
ASCII 码在 32~126 范围内的字符称作普通字符(参见附录 A)，其它字符称作控制字符。
普通字符可从键盘直接输入到程序中，而控制字符要用转义字符的形式输入。
- *2.9 掌握转义字符：\n、\t、\\、\'、\"、\八进制数、\x 十六进制数。
\n 的 ASCII 码为 10，其功能等同于回车键。\t 的 ASCII 码为 9，其功等同于 Tab 键。
在文本输出状态，输出窗口通常为 80 列，等分成 10 区。
- 2.10 如果\与其后字符不组成转义字符，则\无效。
- *2.11 \引导的数默认为八进制，无需 0 打头，\至多与前 3 位八进制数字结合。
- *2.12 如果\引导的是十六进制数，\至多与前 2 位十六进制数结合。

- *2.13 用一对双引号括起来的若干个普通字符或转义字符称作字符串常量，简称字符串。
- *2.14 字符串中字符个数称作字符串长度，简称串长。
- *2.15 字符串按字符的 ASCII 码依次存储，并自动追加 1 个 ASCII 码为 0 的“空字符”'\0' 作为结束标志，因此，字符串的存储量比串长多一个字节。
- *2.16 符号常量名、变量名、函数名、数组名、类型名、类名等统称标识符。
- *2.17 标识符的命名规则为：以字母或下划线打头，由字母、下划线、数字组成。
标识符不要与关键字同名。
- *2.18 符号常量定义格式为：`#define 符号常量 所代表的文本`
其功能是：在预处理时，自动用所代表的文本替换符号常量。
- *2.19 变量必须命名和定义类型(或者说，声明类型)。定义变量类型的语句格式为：
类型名 变量名, 变量名, ...;
- *2.20 定义变量时为其赋值，称作赋初值。
- *2.21 在定义变量时，如果附加了关键字 `const`，所定义变量称作常变量或只读变量。常变量必须赋初值，之后不允许修改其值。
符号常量在预处理时被所代表的文字替换，不参与编译，不占内存。常变量有类型，占内存。
- *2.22 逐步掌握附录 B 中运算的功能、优先级、配对方向、结合方向。
- *2.23 在 C 和 C++ 中，整数的商为整数，舍去余数(不做四舍五入)。
- *2.24 在 C 和 C++ 中，只能用圆括号(所谓的小括号)来规定或改变运算的优先级。圆括号可以嵌套，里层优先。
- *2.25 不同类型数据作运算时，低精度数据自动向高精度转化后再做运算。
在 VC 6.0 中，`char`、`short`、`unsigned short` 型数据均自动转化为 `long` 型后再做运算。精度由低到高排列为：
`long=>unsigned long=>float=>double`。
- *2.26 自增运算“++”和自减运算“--”均是单目运算，只能作用于变量，可作用于左、右两侧，均使变量相应自增或自减 1。作用于左侧时，算术式的值为变量改变后的值，否则为改变前的值。
自增、自减运算均为左配对，与其在变量哪侧无关。运算优先级与其作用于变量的哪侧有关。
- *2.27 强制类型转换运算的格式和功能如下：
格式：(类型名)(表达式)
功能：把指定表达式的值强制转换为指定类型。
- 2.28 如果转换运算所作用的表达式中没有运算或优先级均高于转换运算，可省略表达式的括号。
- 2.29 C++ 中可省略类型名括号，但 C 中不可省。
- *2.30 实数转整数将舍去小数部分(不做四舍五入)，整数转实数将舍去超出数位(做四舍五入)，整数间的转换涉及到补码。
- *2.31 会用类型转换实现“四舍五入”。例如，“`(int)(x*100+.5)/100.`”是保留 x 的 2 位小数。
- *2.32 赋值运算的格式和功能如下：
格式：变量=表达式
功能：把赋值号“=”右侧表达式的值赋给左侧的变量，并以左侧变量为整个赋值表达式的值。
- *2.33 如果赋值号两侧的类型不一致，右侧表达式的值将被自动转换为左侧变量的类型。转换规则与强制类型转换相同。
- *2.34 在 C 和 C++ 中，

$$\square = \square \odot \diamond \quad \text{等价于} \quad \square \odot = \diamond$$

其中， \square 是同一个变量， \diamond 是表达式， \odot 称作复合(或组合)赋值运算符， \odot 是下述运算符之一：

$$+, -, *, /, \%, \gg, \ll, \&, \wedge, |$$

- *2.35 用逗号分隔(或者说连接起来)的一组表达式统称逗号表达式。
格式：表达式, ..., 表达式
功能：从左至右依次计算逗号表达式中各表达式的值，并以最后一个表达式的值的作为整个逗号表达式的值。
- *2.36 将数学式改为 C 表达式时应注意：①乘号*不可省略；②正确改写角码、运算符、括号等；③正确书写函数名；④把握好运算优先级，分数线改为/时，分子、分母相应加括号；⑤正确拆分不等式组；⑥通常，整数除法应改用实数除法。

第 3 章 程序设计初步

- *3.1 赋值表达式后缀分号称作赋值语句。
- *3.2 头文件 `iostream.h`、`iostream`、`iomanip` 均提供了 `cin`、`cout` 等 C++ 标准输入输出流对象，它们的格式和功能如下：
格式 `cout<<表达式...<<表达式;`
功能 从右至左依次计算各表达式，然后，从左至右依次输出各表达式的值。
格式 `cin>>变量...>>变量;`

功能 把输入数据从左至右依次赋给各变量。

<<称作输出运算，如果其后的表达式中有比它优先级低的运算，应适当加括号。

>>称作输入运算。输入数据间用空格、回车键分隔，并用回车键确认所输入数据。

3.3 头文件 `iomanip.h` 提供了许多用于实现特殊输入和输出格式的所谓控制符(参见表 3.1)，它内嵌了头文件 `iostream`。

*3.4 头文件 `iostream` 和 `stdio.h` 均提供了 `putchar`、`getchar`、`printf`、`scanf` 等输入输出函数。`putchar` 和 `getchar` 的格式和功能如下：

格式 `putchar(表达式)`

功能 输出以指定表达式的值为 ASCII 码的字符，并自动追加一个回车符。

格式 `getchar()`

功能 输入一个字符，并以该字符为函数值(字符型)。

注意：`getchar` 可提取空格及回车、Tab 等控制字符，只提取输入中相应位置上一个字符，因此，输入字符间无须分隔，否则也被视为输入字符。

3.5 `printf` 函数的格式和功能如下：

格式 `printf(格式控制串, 输出表列)`

功能 ①从右至左依次计算输出表列中各表达式；②从左至右依次输出格式控制串中的字符，其中，%引导的格式控制符由输出表列中相应表达式的值所取代。

输出表列是一组用逗号分隔的表达式(又称输出项，可以为 0 项)。格式控制串可以是表达式，其中，格式符用于控制输出表列中相应表达式的输出格式(参见 第 04 讲：程序设计初步)。

3.6 `scanf` 函数的格式和功能如下：

格式 `scanf(格式控制串, 地址表列)`

功能 ①从右至左依次计算地址表列中各表达式的值；②从标准输入设备提取数据，并从左至右依次存储到指定的存储单元。

格式控制串可以是表达式。地址表列是一组用逗号分隔的地址，可用是表达式，常用形式为：

&变量

其中&是地址运算符，它表示取其变量的起始存储地址。

`scanf` 可提取空格以及回车、Tab 等部分控制字符，但必须以字符提取(而非字符串)。

`scanf` 函数格式符与 `printf` 函数的格式符的功能相似(参见第 04 讲：程序设计初步)。`scanf` 的合法格式符不分大小写、实型格式符均等效。

*3.7 在 C 和 C++中，逻辑值均为整数，真即 1，假即 0。高版本 C++中新增了逻辑类型 `bool`，它只有 1 和 0 这两个 1 字节整数，可分别表为 `true` 和 `false`。字符和数作为逻辑量时，非 0 即真。指针作为逻辑量时，非空即真。逻辑运算如下表所示：

右元	!右元
假(0)	1
真(非 0)	0

左元	右元	左元&&右元	左元 右元
假(0)	假(0)	0	0
假(0)	真(非 0)	0	1
真(非 0)	假(0)	0	1
真(非 0)	真(非 0)	1	1

*3.8 关系式的值为逻辑值。

*3.9 对于形如

□||□||...

的逻辑式，从左至右计算□的逻辑值，遇真即止。

*3.10 对于形如

□&&□&&...

的逻辑式，从左至右计算□的逻辑值，遇假即止。

*3.11 `if` 语句的格式和功能如下：

格式 `if(条件)语句 1 else 语句 2`

功能 如果条件成立，执行语句 1，否则执行语句 2。

如果语句 2 为空语句，可同 `else` 一起省略。

注意：`else` 不可独立使用，它与其前最近一个尚未配对的 `if` 配对，为避免歧义，通常只在 `else` 中嵌套 `if` 语句。

*3.12 条件表达式的格式和功能如下：

格式 条件?表达式 1:表达式 2

功能 如果条件成立，取表达式 1 的值，否则取表达式 2 的值。

条件表达式可以嵌套，:与其前最近一个尚未配对的?配对。

*3.13 掌握目前所学运算的优先级(见附录 B)。

3.14 switch 语句的格式和功能如下:

格式

```
switch(表达式){  
...  
case 常量 i:语句组 i  
...  
default:语句组 n+1  
}
```

功能 如果表达式的值等于常量 i, 则从语句组 i 开始执行, 否则执行语句组 n+1。

switch() 中表达式的值必须是整数(可以是字符或逻辑值), “default:语句组 n+1” 可缺省, 每个语句组称作一个分支。为使各分支独立, 通常以 break、return、exit 等语句结尾。

*3.15 break 语句的格式和功能如下:

格式 break;

功能 结束本层 switch 语句或本层循环语句。

*3.16 while 语句的格式和功能如下:

格式 while(表达式)循环体

功能 当表达式的值为真时, 重复执行其后循环体。

循环体是循环语句的内嵌语句, 可以是空或复合语句(下同)。

*3.17 do-while 语句的格式和功能如下:

格式 do 循环体 while(表达式)

功能 重复执行循环体, 直到表达式的值为假。

*3.18 for 语句的格式和功能如下:

格式 for(表达式 1;表达式 2;表达式 3)循环体

功能 ①计算表达式 1; ②当表达式 2 的值为真时, 重复执行循环体和计算表达式 3。

表达式 1、表达式 2、表达式 3 均可缺省, 但保留分号。缺省表达式 2 为永真。

*3.19 continue 语句的格式和功能如下:

格式 continue;

功能 跳过本层循环体中剩余语句。

3.20 掌握以下算法: 计算分段函数、打印字符图形、递推(迭代)、累加、阶乘、辗转相除法、穷举(枚举)、判断素数、分解整数因子、分解数字。

*3.21 分段函数的一般形式为:

if(条件 1)f=算式 1;

else if(条件 2)f=算式 2;

...

... else if(条件 n-1)f=算式 n-1;

else f=算式 n;

或: f=条件 1?式 1:条件 2?式 2:…:条件 n-1?式 n-1:式 n

3.22 分支结构的一般格式:

if(条件 1)操作 1;

else if(条件 2)操作 2;

...

... else if(条件 n-1)操作 n-1;

else f=操作 n;

*3.23 输出实心字符图形的一般格式为:

for(i=1;i<=行数;i++){

for(j=1;j<=第 i 行前置空格数;j++)cout<<' ';

for(j=1;j<=第 i 行字符数;j++)输出所用字符

cout<<endl;//结束第 i 行

}

3.24 如果某问题已表述为“当◇时重复执行□”, 即可用下述语句实现:

while(◇)□

3.25 如果某问题已表述为“对于 i=a~b 重复执行□”, 即可用下述语句实现:

for(i=a;i<=b;i++)□ 或 for(i=b;i>=a;i--)□

3.26 如果某问题已表述为“i 从 a 开始, 当◇时重复执行□”, 即可用下述语句实现:

for(i=a;◇;i++)□

3.27 对于已知项数和通项的累加，通用格式为：

和的类型 S=0;

for (i=1; i<=项数; i++) S+=第 i 项;

3.28 对于已知通项和结束条件的累加，通用格式为：

和的类型 S=0;

for (i=1; !结束条件; i++) S+=第 i 项;

3.29 对于已知项数和各项递推式的累加，通用格式为：

和的类型 S=0, T=初始项;

for (i=1; i<=项数; i++)

S+=T, 推下项 T;

3.30 对于已知结束条件和各项递推式的累加，通用格式为：

和的类型 S=0, T=初始项;

for (i=1; !结束条件; i++)

S+=T, 推下项 T;

*3.31 辗转相除法的要点：当 b=0 时，a 和 b 的最大公约数是 a，否则转换为求 b 和 a%b 的最大公约数。

3.32 穷举法又称枚举法，它是在有限或可列集中搜索满足条件的解。穷举法的要点：①确定解的搜索范围，并按某种规律排序(尽可能不重复)；②确定所满足的条件，并在上述搜索范围内求解。

注意：应适当利用条件缩小搜索范围，或缩小搜索范围以减少条件。

*3.33 对于 2 以上的整数 n，n 是素数 $\Leftrightarrow 2 \sim \sqrt{n}$ 中没有 n 的因子。

3.34 对于正整数 n，分解整数因子的过程为：i 从 2 开始，当 i 是 n 的因子时，去除 n 中 1 重 i 因子(n/=i)，否则 i++，直到 n=1 为止。

*3.35 对于非负整数 n，n 的个位数=n%10，n 的十位数=n/10%10，n 的百位数=n/100%10，一般地，n 的 10^k 位数=n/ 10^k %10。

第 4 章 函数

*4.1 从用户使用角度看，函数分为系统函数和用户自定义函数。

*4.2 自定义函数的格式为：

格式 函数值的数据类型 函数名(形参表){函数体}

其中，函数值的默认数据类型为 int，形参要逐个定义数据类型。

*4.3 如果函数没有返回值，函数值的数据类型应定义为 void。

*4.4 return 语句的格式和功能为：

格式一 return 表达式; 或 return(表达式);

格式二 return;

功能 结束函数，返回调用者。格式一带返回值，其数据类型与函数值的类型必须相同或兼容，兼容时，返回值被自动转换为函数值的类型。格式二没有返回值，函数值类型应定义为 void。

*4.5 函数一般调用格式为：

格式 函数名(实参表)

其中，实参按从右至左的次序计算，并传递给相应形参。

*4.6 函数参数的传递方式分为传值和引用。对于前缀&的形参，它与对应实参共用内存，称作引用(C++方有此功能)。对于未前缀&的形参，它与对应实参不共用内存，仅传值。

*4.7 对于无参函数，实参表和形参表的括号仍不可缺省。

*4.8 如果函数调用在其定义之前，调用前应作相应声明，声明格式为：

格式 函数值类型名 函数名(形参表);

其中，形参可省略，但形参的类型不可省略。

*4.9 函数不可嵌套定义(函数定义中嵌套定义其它函数)，但允许嵌套调用(函数定义中调用其它函数或自身)。

*4.10 在函数的定义中直接或间接调用自身，称作函数的递归调用，简称递归。

4.11 递归有三大要点：①递归条件(或回归条件)；②非递归操作(回归操作)；③递归操作。

*4.12 递归函数的核心结构为：

结构一 if(回归条件)回归操作 else 递归操作

结构二 if(递归条件)递归操作 else 回归操作

4.13 C 语言要求，函数中定义变量、数据结构的语句必须在其它语句之前。C++没有此限制。

*4.14 变量的属性分为：①数据类型；②作用域；③存储类别；④存在期。

*4.15 函数形参及函数或复合语句内定义的变量称作局部变量，它从定义处生效，只在该函数或复合语句内有效。因此，不同函数或复合语句中的局部变量可以同名，但同名不同义。

*4.16 在所有函数之外定义的变量称作全局变量，从定义处生效。

4.17 对于函数，之前定义的全局变量均称作外部变量。对于复合语句，之前定义的全局变量和同一函数中之前定义的局部变量均称作外部变量。

*4.18 如果局部变量与外部变量同名，则同名外部变量被屏蔽。

*4.19 局部变量的存储类别分为：动态(auto，又称自动)、静态(static)、寄存器(register)。

*4.20 局部变量存储类别的定义格式为：

格式 存储类别 数据类型 局部变量表；

其中，存储类别和数据类型的位置可交换，默认存储类别为 auto，默认数据类型为 int，但不能同时缺省。

*4.21 auto 型局部变量占用动态数据区，当函数调用结束时释放所占内存。register 占用 CPU 中的寄存器，但寄存器不够用时，占用动态数据区。

*4.22 全局变量和 static 型局部变量占用静态数据区，默认初值为相应数据类型的 0。

*4.23 static 型局部变量的定义和初始化只执行一次，即使离开其作用域也不释放所占内存。

4.24 定义全局变量时，如果加前缀 static，称作内部全局变量，不能被其它文件调用，否则，称作外部全局变量，允许其它文件调用。使用其它文件中定义的外部全局变量，需作 extern 声明。

4.25 定义函数时，如果加前缀 static，称作内部函数，不能被其它文件调用，否则，称作外部函数，允许其它文件调用。使用其它文件中定义的外部函数，需作 extern 声明。

*4.26 预处理命令在程序编译前执行，其主要功能是“文本置换”。每个宏定义必须独占一行。预处理不是语句，不可随意跟分号。

*4.27 不带参数宏定义的格式和功能如下：

格式 #define 宏名 文本

功能 在预处理时，将程序中之后出现的这个宏名均用指定的文本置换。

4.28 带参数的宏定义的格式和功能如下：

定义格式 #define 宏名(形参表) 文本

使用格式 宏名(实参表)

功能 在预处理时，将程序中之后出现的这个带参数的宏均用指定文本置换，其中，形参被相应的实参直接置换(实际上是两次置换)。形参没有类型的概念，没有函数值的概念，对实参不作运算。

4.29 终止宏定义的格式为：#undef 宏名

4.30 允许重新定义宏所代表的文本，新定义只作用于其后的宏名。

4.31 “文件包含”处理(加载文件命令)

格式一 #include<文件名>

格式二 #include“文件名”

功能 把指定文件加载到此处。如果没有指定文件路径，前种格式直接到存放 C 头文件的目录中查找，后种格式先在程序文件所在目录中查找，如果未找着，方到存放 C 头文件的目录中查找。

4.32 条件编译命令的格式和功能如下：

格式一 #ifdef 宏名 程序段 1 #else 程序段 2 #endif

格式二 #ifndef 宏名 程序段 1 #else 程序段 2 #endif

格式三 #if 常量表达式 程序段 1 #else 程序段 2 #endif

功能 如果指定的宏名已定义(格式一)、宏名未定义(格式二)、条件为真(格式三)，保留程序段 1，否则保留程序段 2。最终，整个程序段仅剩程序段 1 或程序段 2。

当程序段 2 为空时，#else 可省略。格式三中的条件必须是常量式。

第 5 章 数组

*5.1 定义数组语句的一般格式和功能为：

格式 数据类型 数组名[第 1 维长度]…[第 n 维长度]

功能 为数组分配相应大小的连续内存，用于依次存储数组元素，并将起始地址赋给数组名。

*5.2 数组的各维长度必须是常量(表达式)，其整数位有效(不作四舍五入)。

5.3 数组的起始地址又称基址。把基址赋给数组名是在分配内存时由系统完成的，之后不允许修改。

*5.4 数组元素又称下标变量，下标变量的使用格式为：

格式 数组名[第 1 维下标]…[第 n 维下标]

*5.5 各维下标均从 0 开始，可用表达式表示，其值的整数位有效(不作四舍五入)。

5.6 下标变量按低维优先顺序存储，对于二维数组又称行优先。

*5.7 下标变量的使用与普通变量基本相同。

5.8 数组也有全局和局部之分，局部数组也有存储类别。

5.9 5.9 C 和 C++编译系统不检测下标越界，越界则顺延至数组所申请的存储空间之外(危险)。

*5.10 定义数组的同时可对其元素赋初值。

格式 数据类型 数组名[第 1 维长度]…[第 n 维长度]={数据表}

功能 定义数组的同时将数据表中数据依次赋给数组元素，未赋值元素的初值为相应数据类型的 0。

*5.11 数据表中允许嵌套数据表，最大嵌套层数不得超过数组维数。数据表和内嵌数据表必须非空，不能“超长”——不允许赋给越界下标变量。

*5.12 对数组赋初值时，可省略第 1 维长度，如果省略，其值为恰好存下数据表所需长度。

*5.13 以字符为数据元素的数组称作字符数组。字符数组可用字符串初始化。

*5.14 对字符串和字符数组可以作输出输入等整体操作，这些操作要求有结束标志'\0'。

*5.15 字符串和字符数组的整体输入语句有：

格式 cin>>字符地址变量表达式

或 scanf("%s", 字符地址变量表达式)

功能 把读入字符依次存储到从指定地址开始的内存中，并自动追加结束标志。读入时，遇到空格、回车、Tab 键等数据分隔符结束。

*5.16 字符串和字符数组的整体输出语句有：

格式 cout<<字符地址表达式

或 printf("%s", 字符地址表达式)

功能 从指定地址开始依次输出字符，直至结束标志。

5.17 原则上，字符数组不要求有结束标志，因此，要相应附加结束标志，方可作整体操作，否则，操作将顺延至字符数组之外。

*5.18 掌握常见的字符串函数。

格式	功能	头文件
strcat(串 1, 串 2)	将串 2 复制、连接到串 1 的第一个'\0'处。	string.h
strcpy(串 1, 串 2)	将串 2 复制到串 1。	
strcmp(串 1, 串 2)	比较串 1 和串 2 的大小。	
strlen(串)	求字符串的长度。	

5.19 了解字符串类 string。

*5.20 掌握冒泡排序、选择排序，了解插入排序、折半查找等。

第 6 章 指针

*10.1 内存的基本单位是字节，每个字节都有相应的编号，称作地址。

*10.2 对任意变量 x，&x 称作变量 x 的指针，变量 x 的数据类型称作&x 的基类型，此&称作取地址运算。

*10.3 变量的指针简称变量指针，它以该变量的基址(变量所占内存的起始地址)为值(而非该变量的值)，又称指向变量的指针。变量指针不是单纯的地址，它蕴涵着基类型(所指向变量的类型)。

*10.4 作为单目运算，&是取地址运算，只能作用于变量(包括常变量)。

*10.5 变量指针不是变量，它所指向变量可表为：

*指针

其中，*称作指向运算。

*10.6 指针也是一种数据类型，其类型名为：

基类型名*

*10.7 在 VC6.0 中，每个指针占 4 字节内存。~~在 C 和低版本 C++ 中，每个指针占 2 字节内存。~~

*10.8 用于存储指针的变量称作指针变量。指针变量的定义格式为：

存储方式 基类型 *指针变量

其中，基类型是指针变量拟指向变量的数据类型，*表示其后的变量是一个指针变量，它不是变量名的组成部分，也不是指向运算。

10.9 指针的值为无符号 16 进制整数，它与数不兼容，除 0 外，不允许把数直接赋给指针变量。

10.10 对任意数组或指针 a 有，&a[i]=a+i，a[i]=(a+i)，特别地，&a[0]=a，a[0]=*a。p+i 相当于向后跳过 i 个数据(而非跳过 i 个字节)，[]称作下标运算或变址运算。由此即可定义下述运算：-、++、--、+=、-=。

*10.11 数组作为函数参数，它只传递数组的基址。下标变量可作为实参，但不可作为形参，实际上，它被误认为数组。

10.12 二维数组的基址以行为基类型。对于二维数组 a，a+i(即&a[i])指向第 i+1 行(或者说，行 i、i 行)，(a+i)(即 a[i])指向 a[i][j]，即，&a[i][j]=*(a+i)+j=a[i]+j，a[i][j]=*(*(a+i)+j)=*(a[i]+j)。

10.13 二维数组的行指针变量的定义格式为：二维数组的类型 (*行指针变量)[列数]

*10.14 指针运算可简单地分为以下 6 类：

(1)单目运算(9 个)：!、&、*、++、--、()、sizeof、delete 和 new(参见 7.1.7)

(2)指针与指针间运算(10 个)：=、-、>、>=、<、<=、==、!=、&&、||

(3)指针与整数间运算(5 个)：+、+=、-、-=、[]

(4)流对象与指针运算(2 个)：>>、<<

(5)结构体指针(参见 7.1.5)或对象指针(参见 9.5)与其成员间运算: ->

(6)合法的三目运算、逗号运算

以上, +、-、++、--、+=、-=均以基类型数据为基本单位。

10.15 函数名是函数的入口, 即, 指向自身的指针, 称作函数的指针, 简称函数指针。

*10.16 函数指针也是一种数据类型, 用于存储函数指针的变量称作函数指针变量, 其定义格式为:

格式 函数值类型(*函数指针变量)(形参类型表)

其中, 类型均是函数指针变量拟指向函数的类型。

*10.17 如果 p 是指向函数 f, 则函数名 f 可用 p 和(*p)表示。

10.18 函数返回值可以是指针, 定义函数的一般格式为:

函数值的基类型*函数名(形参列表){函数体}

10.19 以指针为基类型的指针称作指向指针的指针, 以指向指针的指针为值的变量称作指向指针的指针变量, 其定义格式为:

存储方式 基类型 **指向指针的指针变量

10.20 以指针为元素的数组称作指针数组。