Table of Contents

C程序快速入门

C语言的运算符和表达式

选择结构

循环结构

数组

C语言学习笔记

本书主要时记录我学C语言时所记录的笔记，其中主要是我从参考书上摘录下来的

可以关注我的微信公众号：我的数学世界

我的博客（这个19年底左右就会停）我的数学世界

或者这个博客我的数学世界

powered by Gitbook该文件修订时间： 2019-02-25 21:44:56

C程序基本认识

C程序的基本框架为：

```
#include "stdio.h"
main()
{
    <语句系列>
}
```

其中 `#include "stdio.h"` 为编译预处理命令，位于程序开始，命令后无分号

`main()` 是C程序的主函数，任何C程序有且仅有一个主函数，都从主函数处开始执行，main是函数名，()表示这个函数没有参数，“{”和“}”括起来的部分叫函数体，分别表示函数体的开始与结束

语句用于描述对象以及对对象的操作，语句写在函数体中，以分号结尾

一行可以写一个或多个语句

powered by Gitbook该文件修订时间： 2019-02-25 20:19:04

数据类型等

powered by Gitbook该文件修订时间： 2019-02-26 22:39:57

标识符、变量与常量

1. 标识符

需要人为命名的如：变量名，符号常量名，函数名，数组名，文件名等对象称为变量名

规定：

1. 只能由英文字母、下划线、数字组成，只能以字母或下划线开头，但下划线开头的多为系统定义的变量
2. 变量名区分大小写，但通常用小写（可读性）
3. 标识符的可辨识长度在不同系统有不同规定，最短为8个，最长为32个。一般不超8个
4. 不能用系统保留字（关键字）作为标识符，C语言有32个

2. 变量

程序数据中的两种形式：常量和变量，常量不能改变，变量可变

变量要先声明（定义）后使用且命名遵循标识符的命名规则，定义的任务要包括变量的类型（int，float），变量的名字和初值，例如：`int a , sum= 0;`为定义a，sum变量为整型变量，a的初值不确定，sum初值为0，且通过赋值运算符`=`可以改变变量值

3. 常量

main函数前用`#define PI 3.14`称为宏定义命令，它定义PI代表常量3.14，符号常量不能改变

符号常量还可以用const和类型说明符来声明如：`const float PI = 3.14;`

符号常量通常用大写

powered by Gitbook该文件修订时间： 2019-02-28 15:25:53

数据类型

- 基本类型
  - 整型（int）
  - 字符型（char）
  - 实型（float、double）
- 构造类型
  - 数组
  - 结构体
  - 共用体
  - 枚举型
- 指针类型
- 空指针（void）

1. 整型数据

- 整型常量
  - 十进制整数
  - 0开头的八进制整数
  - 0x开头的十六进制整数
- 整型变量

  [unsigned] int、[unsigned] short [int]、[unsigned] long [int]

  方括号内的部分可以缺省不填

2. 实型（浮点型）数据

- 实型常量
  - 小数形式
  - 指数形式，如123e3、123E3、123E+3、1.23e5都表示123×10^3**e**（**E**）前必须有数，且**e**（**E**）后必须为整数
- 实型变量

  可定义为单精度型 `float` 、双精度型 `double` 、长双精度型 `long double`

较大数和较小数进行加减等运算时会被忽略

3. 字符型数据

- 字符型常量
  - 以单引号括起来的一个字符如 `'a','+','2',' '` 等
  - 以“\”开头的特殊字符（转义字符）有的转为控制码，有的转为字符本身如下：

转义字符	功能
\a	警告响铃
\b	退格

\f	走纸
\n	换行
\r	回车
\t	水平制表
\v	垂直制表
\\	反斜杠
\'	单引号
\"	双引号
\?	问号
\ddd	1~3位八进制数所代表的字符，如'\101'代表'A'
\xhh	1、2位十六制数所代表的字符，如'\x41'代表'A'

- 字符串常量

  用双引号括起来的字符就是字符串常量，如 `"CHINA","a","2"` 等，且内部储存时以 `'\0'` 结尾，常存储在字符数组中

没有字符串变量

> **[danger]** 注意
>
> 没有字符串变量

powered by Gitbook该文件修订时间： 2019-02-27 16:14:15

C程序语句

- 执行语句
  - 表达式语句 `t=a; a++`
  - 函数调用语句 `printf("a=%d,b=%d\n",a,b);`
  - 流程控制语句
- 声明语句
  - `int a,b,t;` 为声明语句，旨在给变量分配空间以方便存储数据
  - `;` 为空语句，即在要求至少有一条语句但不做执行时使用
- 块语句
  - 语法中要求只能有一条语句，但难以用一条语句表达
  - 形成局部化封装体，一般语句块中定义的量仅在语句块中有效

```
#include "stdio.h"
main()
{
    int x = 1;
    {
        int x = 2;
        {
            int x = 3;
            printf("%d\n",x);
        }
        printf("%d\n",x);
    }
    printf("%d\n",x);
}
```

程序的运行结果为

```
3
2
1
```

powered by Gitbook该文件修订时间： 2019-02-28 15:30:56

输出、输入函数

1. 格式输出函数

`printf("格式说明",项目输出表);`

项目输出表是由逗号分隔的0~n个表达式

格式说明：

- 格式字段

  说明项目输出表中各表达式的数据类型、对齐方式、宽度、精度

- 控制字符
- 需要原样输出的字符

格式字段的形式为 `%[对齐方式] [宽度和精度说明] 格式符` 格式符和%之间不能插入其他字符

printf中常用格式符：

格式符	输出项形式
d,i	十进制整数（正数不输出符号）
x,X	无符号十六进制整数，用X时输出十六进制的大写字母
o	无符号八进制整数
u	无符号十进制整数
f	以小数形式输出实数，隐含输出六位小数
e,E	以指数形式输出实数，用E时指数用E表示
g,G	选用%f或%e格式中输出宽度较短的一种格式，用G时指数用大写表示
c	输出一个字符
s	字符串

> **[info]** 注
>
> 整数部分表达宽度小数部分表达精度 `%5.2f` 表示宽度为5个字符位，精度为保留两位小数

另：

> **[warning]** 注意
>
> 宽度说明缺省或小于实际宽度时按实际宽度输出
>
> 精度说明缺省时小数部分取六位

默认情况下数据按右对齐输出，当在 `%` 后使用 `-` 时则按左对齐输出

在右对齐时宽度说明前加 `0` 则数据前多余的空位用“0”补齐

2. 格式输入函数

`scanf("格式说明",地址表列);`

地址表列指键盘输入的数据应存入的变量地址（&变量），不能是表达式

> **[info]**注
>
> 除格式符以外的字符必须按原样输入，一般是输入的间隔字符

如：

```
scanf("a = %d,b = %d",&a,&b);
```

输入的必须是

```
a = 2,b = 3
```

格式说明中指定列宽，系统会按列宽自动截取数据

若使用`*`而不是`&`则将在输入数据时跳过一项，不赋值给任何变量，且不应指定精度

powered by Gitbook该文件修订时间： 2019-02-27 17:27:23

算术运算

powered by Gitbook该文件修订时间： 2019-02-28 15:25:46

运算符与表达式

学习运算符时应注意：

1. 运算符的功能
2. 运算符对运算对象的限制
   - 运算对象个数
   - 操作数
   - 运算对象的类型
3. 表达式值的类型
4. 运算符的优先级
5. 结合方向
   - 左结合
   - 右结合

C语言运算符的优先级与结合性

优先级	运算符	结合方向
1	() [] .（取成员）->（指向成员）++ --（后缀）	→（左结合）
2	!（逻辑非）~（按位取反）++ --（前缀）-（负）*（间接引用）&（取地址）sizeof（容量运算）	←（右结合）
3	* / %	→
4	+ -	→
5	<<（左移运算）>>（右移运算）	→
6	< <= > >=（关系运算）	→
7	== !=（关系运算）	→
8	&（按位与）	→
9	^（按位异或与）	→
10	\	（按位或） →
11	&&（逻辑与）	→
12	\	\ （逻辑或） →
13	? :（条件运算）	←
14	= += -= *= /= %= <<= >>= &= ^= \	=（赋值运算） ←
15	,（逗号运算）	→

powered by Gitbook该文件修订时间： 2019-02-28 15:59:48

算术运算符和算术表达式

单目运算符：-（负）+（正）

双目运算符：*（乘）/（除）%（取余）+（加）-（减）

- 单目运算符是右结合，双目运算符是左结合
- 单目运算符优先级高
- 算数运算符要求的操作数为数值型（整型、实型、字符型）
- %要求的两个操作数必须为整型
- 运算对象可以是常量、变量、函数等

赋值运算

powered by Gitbook该文件修订时间： 2019-02-28 16:17:48

赋值运算符

`=` 的作用是将右边表达式的结果存入左边的变量，且左边是合法的变量名

一般形式为： `<变量> = <表达式>`

> **[info]** 注 以下表达式均合法：
>
> ```
> int a,b,c;
> a = b = 5;
> a = 6 + ( c = 6 );
> a = ( b = 5 )+( c = 6 );
> ```

在 `=` 前加上二元运算符就变成复合的赋值运算符，如 `+= -= *= /=`

```
a += 3; /*相当于a = a + 3;*/
x *= y + 8; /*相当于x = x * (y + 8);*/
```

powered by Gitbook该文件修订时间： 2019-02-28 19:55:17

自增1（自减1）运算符

`x += 1;` 和 `x++;` 等价

`++和--` 叫自增1和自减1运算符

前缀和后缀的区别：

```
y = x++; /*相当于y = x; x += 1;*/
y = x--; /*相当于x += 1; y = x;*/
```

> **[danger]** 注意
>
> 如果有
>
> ```
> int i = 5;
> printf("%d",i++);
> ```
>
> 在不同的系统中会输出不同的结果，为6或5，所以最好不要这么写

powered by Gitbook该文件修订时间： 2019-02-28 20:09:44

类型转换等运算

powered by Gitbook该文件修订时间： 2019-02-28 20:11:44

类型转换

1. 自动类型转换

- 非赋值运算的类型转换

  运算中char，short型都将转换成int型，所有unsigned short型转换成unsigned型，所有long型转换成unsigned long型，所有float型转成double，若数据类型仍不一致，则低级（占字节少的类型）向高级（占字节多的类型）转换

- 赋值运算的类型转换

  赋值运算符两边数据类型不同时，右侧向左侧转换

  例如

  ```
  int a,j,y;
  float b;
  long d;
  double c;
  y = j + 'a' + a * b - c / d;
  ```

  其运算顺序和隐含的类型转换如下：

  1. 计算 `a*b` ，结果为double型
  2. 计算 `c/d` ，结果为double型
  3. 计算 `j+'a'` ，结果为整型
  4. 将1和3的结果相加，结果为double型
  5. 将第4步结果减第2步的结果，结果为double型
  6. 给y赋值，先将第5步的结果double型转成整型，截掉double的小数部分，压缩成int型

2. 强制类型转换

一般形式为： `（类型名）（须强制转换的表达式）`

如：

```
(double)b * 3; /*将b的值转换为double型*/
(int)(x) % a; /*将x的值转换成int型，再做%运算*/
(float)(5 % 3); /*将5%3的结果转换为float型*/
```

> **[warning]** 注意
>
> 类型转换是对值而不是对变量，如：
>
> `(double)x = 7;` 变成了 `((double)x) = 7;` ，错误
>
> 类型转换是一元运算符，优先级与一元运算符相同

powered by Gitbook该文件修订时间： 2019-02-28 21:25:38

逗号运算符

逗号运算符又叫“顺序求值”运算符，它用逗号连接若干表达式顺序求其值

一般形式为：

`表达式1,表达式2,…,表达式n`

逗号表达式先求表达式1的值，再求表达式2的值，整个表达式的值是最后一个表达式的值，且逗号运算符的优先级最低，如：

`a = 3 * 5, a * 4;`

第一表达式 `a = 3 * 5` 的值是15，第二表达式 `a * 4` 的值是60，整个表达式的值是60

`,` 除做运算符以外，还常做分隔符

如：

```
int a,b,c;
printf("%d,%d,%d\n",a,b,c);
```

其中 `,` 作为分隔符，若第二行改为 `printf("%d",(a,b,c))` ，则 `(a,b,c)` 为逗号表达式，值为c的值

容量运算符

一元运算符sizeof的使用形式为：

```
sizeof(目标)
```

其功能是返回该目标所占的字节

目标可以为：

1. int，float一类的类型符
2. `a+b` 形式的表达式
3. 数组或结构

> **[danger]** 注意
>
> sizeof()不是函数，是运算符，结果是无符号整数！
>
> sizeof()不是函数，是运算符，结果是无符号整数！
>
> sizeof()不是函数，是运算符，结果是无符号整数！

powered by Gitbook该文件修订时间： 2019-02-28 21:40:46

位运算符

位运算符只能作用于整数分量，即有无符号的char，short，int，long型

除~是一元运算符外，全为二元运算符

位运算符：

运算符	功能	
&	按位与	
\		按位或
^	按位异或	
<<	左移	
>>	右移	
~	求反	

1. 按位与（&），按位或（|），按位异或（^）

   前三个运算分别对二进制每一位按照真值表运算

   &：当两个都是1时返回1，否则返回0

   |：当两个都是0时返回0，否则返回1

   ^：当两个都为0或都为1时返回0，否则返回1

2. 按位左移（<<），按位右移（>>）

   将二进制位左移或右移若干位，左移一位相当于乘2，右移一位相当于除2

   左移时右边的空位用0填充

   右移时若右移无符号量则左边用0填充，若右移有符号量则左边空出的位数用符号位填充（算术移位）或用0填充（逻辑移位）

3. 按位求反（~）

   求整数的反码，即将0变为1，1变为0

关系运算符和逻辑运算符

1. 关系运算符和关系表达式

关系表达式是用关系运算符和括号将运算对象（常量、变量、函数等）连接起来的式子

关系运算符共有六种：

`< <= > >= == !=`

前四种运算符 `< <= > >=` 运算级相同，后两种运算符 `== !=` 运算级相同，且前四种高于后两种

关系运算符优先级低于算术运算符但高于赋值运算符

> **[warning]** 注意
>
> 若关系表达式成立，则表达式的值为“真”，用整数“1”表示
>
> 若关系表达式不成立，则表达式的值为“假”，用整数“0”表示

2. 逻辑运算符和逻辑表达式

逻辑运算符：

```
&& （逻辑与）
|| （逻辑或）
!  （逻辑非）
```

优先级：

`!`>算术运算符和关系运算符> `&&` > `||`

逻辑运算真值表：（真为T，假为F）

a	b	!a	!b	a&&b	a\	\	b
T	T	F	F	T	T		
T	F	F	T	F	T		
F	T	T	F	F	T		
F	F	T	T	F	F		

- 逻辑运算符中所有非零数值均认为真
- 逻辑运算中并非所有运算符都会被执行
  1. `a&&b&&c` 中当a为真时才判断b的值，当a和b均为真时才判断c的值
  2. `a||b||c` 中a为假时才会判断b的值，当a和b均为假时才判断c的值

3. 条件运算符和条件表达式

条件表达式的格式为：`表达式1?表达式2:表达式3`

求解过程为：

判断表达式1的值，若为真，则执行表达式2，否则执行表达式3

if语句

1. 简单选择结构

语句形式为：

```
if(表达式)
    语句;
```

2. 二路选择结构

语句形式为：

```
if(表达式1)
    语句1;
else(表达式2)
    语句2;
```

3. 多路选择结构

```
if(表达式1)
    语句1;
else if(表达式2)
    语句2;
……
else if(表达式n)
    语句n;
else
    语句n+1;
```

> **[warning]** 注意
>
> 这一节并没记完，因为我觉得if嵌套什么的太简单了，没得说（我就是懒得记），实在想看就给我说或者看我心情

powered by Gitbook该文件修订时间： 2019-03-01 17:03:40

switch语句

一般形式：

```
switch(表达式)
{
    case 常量表达式1: 语句1;
        break;
    case 常量表达式2: 语句2;
        break;
    …
    case 常量表达式n: 语句n;
        break;
    default: 语句n+1;
}
```

在执行时先计算switch后的表达式的值，再与1~n个常量表达式的值进行比较，当表达式的值与某个case后常量表达式的值相等时，则执行该case后的语句，然后执行break语句跳出switch结构，若所有常量表达式的值都不等于switch后表达式的值，则执行default后的语句

注意：

1. case后的常量表达式的值应与switch后的表达式的值类型一致且都为整型或字符型
2. 同一个switch语句中所有case后常量表达式的值必须互不相同，否则会发生冲突
3. 当表达式的值与某个case后常量表达式的值相等时，则执行该case后的语句，然后执行break语句跳出switch结构，若所有常量表达式的值都不等于switch后表达式的值，则执行default后的语句
4. 各case和default顺序可以改变，但default前置时要加 `break;` ，最后的语句可以不加 `break;`
5. 多个case可以用一组执行语句，直到遇到 `break;` 才停止，否则执行其后所有case（default）的执行语句
6. 每个case后可有任意多个语句，且无需使用花括号

powered by Gitbook该文件修订时间： 2019-03-01 18:42:58

while循环

语句形式：

```
while(表达式)
    语句;
```

执行过程：

当表达式的值为真时，执行循环体一次，再判断表达式的值，并重复上述过程，直到表达式的值为假时才结束循环

说明：

- 特点为：先判断表达式，再执行语句
- 若循环体包含一个以上语句，应用花括号括起来，以复合语句形式出现，若不加花括号，则只执行到第一个花括号处结束

do-while循环

一般形式：

```
do
    语句;
while(表达式);
```

执行过程：

先执行一次指定的循环体语句，然后判别表达式，当表达式的值为非零（真）时，返回重新执行循环体语句，直到表达式的值等于0为止

for循环

一般形式：

```
for(表达式1;表达式2;表达式3)
    语句;
```

说明：

1. 表达式1：一般为赋值表达式，为循环控制变量赋初值
2. 表达式2：一般为关系表达式或逻辑表达式，作为控制循环结束的条件
3. 表达式3：一般为赋值表达式，为循环控制变量增量或减量

`for(i = 1,sum = 0;i <= 100,i++)sum = sum + i` 相当于

```
i = 1,sum = 0;
while(i <= 100)
{
    sum = sum + i;
    i++;
}
```

说明：

1. for语句一般形式中表达式1可以省略，但 ; 不能省略，次时应在for前给循环变量赋初值
2. 若表达式2省略，则默认始终为真
3. 表达式3也可省略，但需自行设置终止条件
4. 可以省略表达式1和3
5. 可以均省略，此时循环不会退出
6. 表达式1和3可以是任意表达式
7. 表达式2可以是关系表达式、逻辑表达式、数值表达式、字符表达式

break和continue语句

`break;`

终止switch或循环语句的执行，跳出当前break所在的控制结构，转去执行后继语句

1. 只能在循环体内和switch语句体内使用break语句
2. 在多重循环嵌套中，break语句仅跳出它所在的那一层循环结构，不能跳出（或终止）多层循环结构

`continue;`

终止本次循环的执行，直接开始下一次循环的执行

> **[warning]** 注意
>
> 1. continue语句只能用于循环语句
> 2. 在（do-）while中continue立即去检查循环控制表达式，在for中立即转向执行表达式3

powered by Gitbook该文件修订时间： 2019-03-01 21:04:09

一维数组

1. 定义

一般形式：

`类型名 数组名[常量表达式]`

其中：

1. 数组的命名符合命名规则
2. “类型名“指数组存储数据的类型，可以为int、float、char等，也可以为指针或结构体等
3. ”[常量表达式]“用来指定数组的长度

2. 一维数组的存储

以 `int a[13]` 为例，元素下标为0~12，元素分别为a[0]，a[1]，…，a[12]共13个元素

3. 一维数组元素的引用

引用形式为：

`数组名[下标值]`

其中：

1. 下标值必须在下标值范围内
2. 下标值形式可以为常量、变量、表达式
3. 数组名不代表任意一个元素，例如a不代表a[]中任意一元素
4. 使用数组元素时可以把他们看作同类型变量使用

4. 一维数组的初始化

一维数组的初始化有两种方法：

1. 给数组的全部元素赋初值

   `int a[5] = {0,1,2,3,4};`

   在给定全部元素赋初值的情况下也可以

   `int a[] = {0,1,2,3,4};`

2. 给数组中的部分元素赋初值

   `int a[5] = {0,1,2};`

   该语句把0，1，2赋给a[0]~a[2]三个元素，其余默认赋初值为0

5. 一维数组的输入、输出

对于一维数组的输入与输出要用循环语句，例如：

```
# include<stdio.h>
main()
{
```

```
    int a[5];
    int i;
    for(i = 0;i < 5;i++)
        scanf("%d",&a[i]);
    for(i = 0;i < 5;i++)
        printf("%d",a[i]);
    printf("\n");
}
```

powered by Gitbook该文件修订时间： 2019-03-02 17:29:02

二维数组

1. 定义

一般形式为：

`类型名 数组名[常量表达式1][常量表达式2]`

其中二维数组命名规则、类型名与一维数组相同，但[常量表达式1]用来指定二维数组的行数，[常量表达式2]用来指定二维数组的列数

2. 二维数组元素的引用

引用形式为：

`数组名[下标值1][下标值2]`

其余特性与一维数组相似

同样的，还可以定义多维数组：

```
int a[4][5][6];
```

3. 二维数组的初始化

二维数组的初始化有两种方法：

- 对数组的全部元素赋初值：
  1. 整体赋初值：

     ```
     int a[3][4] = {1,2,3,4,5,6,7,8,9,10,11,12};
     ```

  2. 分行赋初值：

     int a[3][4] = {{1,2,3,4},{5,6,7,8},{9,10,11,12}};

     在对全部元素赋初值时，第一维的长度可以省略，如：

     int a[][4] = {{1,2,3,4},{5,6,7,8},{9,10,11,12}};

- 对数组的部分元素赋初值

  部分赋值是对数组中各行对应位置的元素赋初值，没有指定元素则自动赋值为0

  在对数组的部分元素赋初值时同样可以省略第一维长度的指定

4. 二维数组的输入/输出

二维数组的输入输出需要两重循环，例如：

```
# include<stdio.h>
main()
{
    int a[3][2];
    int i,j;
    for(i = 0;i < 3;i++)
        for(j = 0;j < 2;j++)
```

```
            scanf("%d",&a[i][j]);
    for(i = 0;i < 3;i++)
    {
        for(j = 0;j < 2;j++)
            printf("%d",&a[i][j]);
        printf("\n");
    }
    printf("\n");
}
```

powered by Gitbook该文件修订时间： 2019-03-02 21:25:44

字符数组

powered by Gitbook该文件修订时间： 2019-03-02 19:51:06

字符数组

1. 字符数组的定义

字符数组定义方法与一维数组类似，例如

```
char string[100];
```

同样可以定义二维字符数组

> **[warning]** 注意
>
> 字符数组和前面一维二维数组基本上都一样，没什么好写的，暂时略过，有需求再加

另：

> **[danger]** 注意
>
> 对于字符串的输入输出可用"%s"而不是"%c"
>
> 对于含空格的字符串的输入应用多个字符数组，如输入"I am a student"则应：
>
> ```
> char str1[8],str2[8],str3[8],str4[8];
> scanf("%s%s%s%s",str1,str2,str3,str4);
> /*这里不用加&，因为字符数组名代表了字符数组的起始地址*/
> printf("%s%s%s%s",str1,str2,str3,str4);
> ```

对于此类问题的解决方案是：

> **[success]** 利用字符串输入/输出函数对字符数组进行输入输出
>
> 1. 字符串输入函数gets()的一般形式为：
>
>    ```
>    gets(字符数组)
>    ```
>
>   例如：
>
>    ```
>    char string[50];
>    gets(string);
>    ```
>
> 2. 字符串输出函数的puts()一般形式为：
>
>    ```
>    puts(字符数组)
>    ```
>
>   例如：
>
>    ```
>    char string[50];
>    gets(string);
>    puts(string);
>    ```

另：

> **[warning]** 注意
>
> 在使用`gets()`，`puts()`函数时只需提供字符串数组名即可