- C语言的基本元素
  - 字符集及词法元素
  - 关键字和标识符
  - 基本数据类型
    - 数据类型概论
    - char类型
    - 整型类型
    - 浮点类型
  - 常量与变量
    - 整型常量
    - 浮点类型常量
    - 字符常量
    - 字符串常量
    - 符号常量
    - 变量声明
  - 运算符和表达式
    - 运算符相关概念
    - 算术运算符
    - 关系运算符
    - 逻辑运算
    - 自增自减运算
    - 赋值运算
    - 逗号运算
    - sizeof运算
  - 位运算
    - 整数的机内表示
    - 位逻辑运算
    - 移位运算
  - 类型转换
    - 类型转换的规则
    - 类型转换的方法
  - 枚举类型
    - 枚举类型的声明
    - 枚举类型定义符号常量
    - 枚举变量的定义

C语言的基本元素

字符集及词法元素

- C语言源程序由字符序列构成,C语言字符集是7位ASCII码的子集
- 编译器对C程序中的字符序列进行词法分析,按规则将其分解为**记号**.
  记号是是程序中具有语义的最基本组成单元.
  记号分为:标识符,关键字,常量,运算符和标点符号.
  编译器从左到右收集字符,尽量建立起最长的记号.

关键字和标识符

- 关键字是被系统赋予特殊含义并有专门用途的字
- 标识符用来给变量,函数,宏和标号等命名.
  标识符由字母,数字,下划线组成,但首字符必须是字母或下划线,不能是数字.

基本数据类型

数据类型概论

数据类型分为：**基本类型，导出类型，空类型**
基本类型包括：字符型，整型，浮点型，布尔型和复数类型
导出类型由基本类型按照一定规则构造而成，包括：数组，指针，结构和联合
空类型有三种用途：1.表示函数不返回任何值;2.说明函数无参数;3.表示指针类型不确定

类型名称	长度	取值范围
signed char	1	-2^7~2^7-1
unsigned char	1	0~2^8-1
signed int/int	4	-2^{31}~2^{31}-1
unsigned/unsigned int	4	0~2^{32}-1
short/short int	2	-2^{15}~2^{15}-1
unsigned short	2	0~2^{16}-1

类型名称	长度	取值范围
long/long int	4	-2^{31}~2^{31}-1
unsigned long	4	0~2^{32}-1
long long	8	-2^{63}~2^{63}-1
unsigned long long	8	0~2^{64}-1
float	4	3.4e-38~3.4e+38(*7位有效数字*)
double	8	1.7e-308~1.7e+308(*15位有效数字*)

关键字signed可以省略,对任何整型默认有符号;int前面有其他修饰符时,int可以省略.

0:00000000
127:01111111
-128:10000000,是00000000的补码
-127:10000001,是11111111的补码

char类型

char类型用于储存字母,数字,标点符号之类的字符,字符数据通过编码在内存中以二进制数储存.
ASCII码采用单字节编码,用指定的二进制组合表示可能的字符.
字符实际上以整数形式储存,所以可以用整数对字符变量赋值.
单引号是**字符常量**的界定符,编译器将'a'转换成对应的编码值,'a'不能写成a,否则被视为标识符.
字符数据的储存形式和整数的储存形式类似,使得字符数据和整型数据可以通用.

整型类型

int是基本的整型类型,long和short是其基础上的拓展,64位环境下int的机器字长为8字节.
输出int类型时可以用 `%d` 说明符,输出unsigned类型数据使用 `%u` 说明符.short类型用 `%hd` ,long类型用 `%ld` ,unsigned short用 `%hu` ,unsigned long用 `%lu` .

浮点类型

精度描述了浮点值中有意义的十进制位个数,范围描述了浮点变量能表示的正的最大浮点值和最小浮点值.
十进制浮点数转换成二进制数:整数部分除二取余,逆序排列;小数部分,乘二取整,顺序排列.
二进制浮点数**$V=(-1)^S*M*2^E$**.其中S是符号位,M是尾数,表示有效数字,E是阶码,表示数位.

计算机用二进制形式储存S,M,E.储存区域分为**符号区,指数区,尾数区**.
对于float型数S一位,E八位,M二十三位;对于double型数,S一位,E十一位,M五十二位.
尾数M用原码表示,小数点前的1可以省略,从而表示范围比实际存储多一位.
阶码E用移码表示,保证阶码可以为负数,float移码为127,double移码为1023.
尾数位数决定了值的精度,阶码位数决定了值的范围,float精度为**7位**,double精度为**15位**.
在可表示范围内,整型一定是精确的,而浮点数只是近似精确的.例如出现尾数区不够,上溢或者下溢的情况.

常量与变量

整型常量

- 语法格式:[前缀]整数部分[后缀]
  - 前缀表示进制:**0(八进制);0x,0X(十六进制)**.
    十进制转换为N进制:除N取余,逆序排列
  - 后缀表示整数类型:u或U表示unsigned;l或L表示long;ul或UL表示unsigned long

浮点类型常量

- 语法格式:[整数部分].[小数部分][后缀]
  - 整数部分和小数部分可以没有,但不能同时没有.
  - 没有后缀是表示double类型,f或F指定float常量.
    E.G.
    12.:double型浮点数;
    .5f:float型浮点数
- 语法格式:[整数部分][.][小数部分]e[+或-]n[后缀]
  - 小数点可以没有,整数和小数部分不能同时没有
  - n是一个整数,表示阶码
    E.G.
    2.E-123L:long double型浮点数;
    10e10:double型浮点数.

字符常量

字符常量指用一对**单引号**括起来的一个字符,形式为:**'字符'**
单引号是字符常量的标志,称为**定界符**.字符常量在机内的值是一个整数,是在机器字符集中的字符码.'0'的值是48,'a'的值是97.
单引号里的字符有两种表示方式:

- 用**字符的图形符号**表示一个字符.图形符号指可打印字符,这只适用于部分字符码为32~126的字符.
  - ' '是空格符,值为32;
  - 单引号字符和反斜线字符必须用转义字符表示,即 `'\''` 和 `'\\'` ;
- 用**转义字符**表示一个字符
  - **字符转义序列**用于表示字符码为0~31的字符和一些特殊字符.
    E.G.
    '\n'表示换行字符,值为10;
    '\0'表示空字符,值为0,通常用来表示一个字符串结束;
    '\\'表示反斜线,值为92;
    '\"'表示单引号,值为39.
  - **数字转义序列**:反斜线后面跟一个字符的**八进制或十六进制字符码**，即\ooo或\xhh
    - 其中ooo表示1~3个八进制数字;hh表示1~2个十六进制数字.
      例如 `'\t'`等价于`'\11'`等价于`'\011'`等价于`'\x9'`等价于`'\x09'` ,它们的值都相同;
    - `'"'` 等价于 `'\"'` 等价于 `'\042'` 等价于 `'\x22'` ;
    - `'\''` 等价于 `\047` 等价于 `\47` 等价于 `\x27` ; `'\\'` 等价于 `'\134'` 等价于 `'\x5c'` .

字符串常量

C语言中没有字符串类型,但可以表示字符串常量,字符串变量则是用字符数组表示.
字符串常量(字符串)是用一对双引号括起来的多个字符的字符序列,形式为:"字符序列"

- 双引号是字符串的定界符,其中的字符可以是图形符号也可以是转义序列.
- ""(无空格)是一个包含0个字符的空字符串.
- 字符串机内储存时,系统自动在其末尾添加一个空字符'\0'作为结束标志,因此字符串储存长度比实际长度大1.
- "A"与'A'完全不同.后者是一个字符常量,储存长度是1,值为65;前者是储存长度为2的字符串常量,本质上是一个包含字符'A'和'\0'的**字符数组**.
- 字符串中的双引号字符必须用转义字符表示;单引号则无所谓.
- 字符串不能中途换行,必须用反斜线作为续行符或者将字符串分段.

符号常量

一个符号常量就是一个标识符,有三种定义符号常量的方法:

1. 用预处理指令#define;
   i. 语法格式:**#define 标识法 文字常量** */*末尾没有分号*/*
   ii. 这种替换在编译之前进行,指令放在文件顶端
   iii. 例如数字转义序列依赖于字符编码方式,不可移植,可将其定义为符号常量.
2. 用类型限定符const;

   i. const称为**类型限定符**,语法格式:**const 类型区分符 标识符=常量;**/*末尾有分号*/;
   ii. 类型相同的标识符可以在一个声明语句中定义,用逗号隔开;
   iii. const声明的标识符必须定义时初始化,编译时会根据定义的类型为其分配相应大小的储存单元,并将值放入其中;
   iv. 对该储存单元只能进行读操作,该标识符是一个只读变量;
   v. const和#define定义的符号常量在实现上有本质的不同,前者定义的标识符**有对应的储存空间**,程序中每出现该标识符都是对存储单元的**访问**(只读).后者定义的标识符**没有对应的存储单元**,编译前由预处理器进行简单的**文本替换**.
3. 用enum枚举类型.

变量声明

变量本质上代表计算机内存中的某一存储单元,可以将值放入其中并访问它.C程序中的一切变量都要遵循**先声明后引用**的原则,以便编译器为变量分配适当长度的存储单元,以确定变量允许的运算.
除了**外部变量**声明有定义性声明和引用性声明外,其他变量的声明都是**定义性声明**.
定义性声明和引用性声明的本质区别在于:前者为变量分配存储单元.后者不分配存储单元.
这里介绍==局部于函数体内的且缺少存储形式的基本类型变量(**自动变量**)==的声明形式.
变量声明的一般形式是:**类型区分符 变量表;**.

- 变量表可以由一个标识符或者由逗号分隔的多个标识符构成.
- 给变量设置初值的方法有两种
  - 通过赋值语句置初值;
  - 变量声明时设置初值.
  - 对于自动变量两种方式的实质一样.**类型区分符 变量=表达式;**
  - 对于自动变量,初值表达式不限于常量表达式,例如:**int count=sum=0;**

运算符和表达式

运算符相关概念

1. 运算符按功能分为:算术运算符,关系运算符,逻辑运算符,按位运算符,赋值运算符,自增自减运算符和逗号运算符;此外,数组下标[],函数调用(),指针间接访问*,结构成员访问.和->,类型强制符都是运算符.
2. 操作数的个数和类型要求:
   i. **单目运算符**只有一个操作数,如取地址运算符&;
   ii. **双目运算符**有两个操作数;
   iii. **三目运算符**有三个操作数,如条件运算符"?:";
   iv. 运算符都对操作数类型有要求,如求余数运算符%的操作数要求为整型.

3. 运算结果的类型.表达式的类型由运算符和操作数决定.当两个不同类型的操作数进行运算进行运算时,会引起数据类型的转换.
4. 运算符有**优先级和结合性**.当表达式中有多个运算符时,运算执行的顺序是由运算符的优先级和结合性决定的.结合性分为**左结合和右结合**.
    i. 优先级高的运算先执行;
    ii. 处于同一优先级的运算符的运算顺序就该优先级运算符的结合性决定.

算术运算符

- /(除)处理负数时, 结果总是**向0取整**;
- %(求余数)处理负数时,结果的符号总是**和左操作数相同**.
  例如-7/2结果为-3;
  -22%-7结果为-1;22%-7结果为1.

关系运算符

关系运算的结果类型是int,值为0或1.由于关系运算符的结合性是左结合,因此数学不等式中的x>y>z应当写为 `x>y&&y>z` .

逻辑运算

逻辑运算有三个运算符:**逻辑与(&&),逻辑或(||),逻辑非(!)**;
逻辑运算的操作数是值为0或非0的表达式.C语言中0值视为逻辑假,0,0.0,'\0',空指针NULL都视为假.
逻辑运算的结果类型是int,值为0或1.
E.G.

- 字符c的值是英文字母则表达式结果为1,否则结果为0: `c>='a'&&c<='z'||c>='A'&&c<='Z'` ;
- 某年是闰年则结果为1否则结果为0: `!(year%4)&&year%100||!(year%400)1` .

自增自减运算

`++` 使内存中储存的变量值加一; `--` 使内存中储存的变量值减一.结果的类型与操作数类型一致.

- `++x` 是先将x的值加1,再以x的新值作为该表达式的值;
- `x++` 是先以x的值作为该表达式的值,然后x再加1;
- **序列点**是程序执行中的一个点,在该点之前所有变量值的改变操作必须完成.定义序列点是为了消除编译器解释表达式时的歧义.下面是定义的序列点:
    - **&&**,**||**,**?:**,**逗号运算符的第一个操作数结尾处**;
    - **完整表达式的结束**,包括表达式语句的**分号处**,do,while,if,switch,for语句的表达式的右括号处,for语句的两个分号处,return语句对表达式末尾分号处.

E.G. `int a=1,b=0`

- `a--&&(a+b)` ,C语言规定运算符&&,||和,**从左到右求值**,&&是序列点,因此表达式执行为1&&0
- `b++?b:-b` ,执行为0?1:-1
- `b+++b++` ,这个表达式的值是不确定的,与编译器有关,因为其中不包含序列点

赋值运算

赋值运算时把数据存入操作数对应的存储单元中.

- **简单赋值**的表达式形式为:操作数1 = 操作数2
  - 先计算右边操作数2的值,再将其赋给左边操作数1对应的存储单元.左值表达式包括:变量名,下标表达式,指针间访表达式,结构成员选择表达式等;
  - 左操作数和右操作数的类型可以不同,赋值前右操作数被自动转换成左操作数类型,表达式的值和类型与左操作数相同.
- **复合赋值**的表达式形式为:操作数1 op= 操作数2
  - op表示+,-,/,%,<<,>>,&,^和|中一个.其可以理解为为:操作数1 = 操作数1 op(操作数2)
  - 区别在于:前者操作数1只计算一次,后者操作数1计算两次
  - `s[i++] += 1` 不等价于 `s[i++] = s[i++]+1` .前者i自增1次,后者i自增2次.且前者相当于直接在 `s[i]` 上加上1,后者相当于将 `s[i]+1` 的值赋给 `s[i+1]` .
  - `i+=2` 可以读作"把2加到i上"; `i=i+2` 可以读作"取i,加上2,再把结果放回i中".

逗号运算

优先级最低的运算符是逗号运算符,它是顺序求值运算符.
一般形式为:操作数1,操作数2,...,操作数n.
它的值是最后一个操作数n的值,其适用于程序中需要执行多个表达式,而语法上只允许一个表达式的情况.
例如计算1累加到10, `for(sum=0,i=1;i<=10;sum+=i,i++);` 不能写成 `for(sum=0,i=1;i<=10;i++,sum+=i);`

sizeof运算

sizeof用于给出一个**数据类型或数据对象所需的字节数**.它有两种形式:

- **sizeof(类型名)**.给出指定数据类型占用的存储字节数;
- **sizeof 表达式**.给出表达式结果的类型占用的存储字节数.
  - `sizeof a+b` 与 `sizeof(a+b)` 不同,前者表示==先求 `sizeof a` ==再加b;
  - sizeof表达式是一个**常量表达式**,其运算是在编译时执行的.因此,当sizeof操作数是表达式时,在编译时分析表达式类型,**运行时不对表达式求值**.

位运算

整数的机内表示

正数在机内是以原码表示的,负数在机内是以补码表示的.
补码变原码:符号位除外各位取反,取反后加一

位逻辑运算

1. **按位取反:~**
   单目运算符~将操作数在内存中的每个二进制位取相反值.
2. **按位与:&**
   i. &运算对两个操作数的逐个二进制位进行与运算.
   ii. 0xff00称为二字节的**逻辑尺或屏蔽码**,它通过&运算保留整数高八位,低八位全部置零.按位与可以用于屏蔽某些位.
   iii. 判断一个整数是否为偶数: `!(x&1)` .
3. **按位或:|**
   i. |运算对两个操作数的逐个二进制位进行或运算.
   ii. 逻辑尺0xff00通过|运算保留了整数的低八位,高八位全部置为1.按位或可以用于打开某些位
4. **按位加:^**
   i. ^运算对两个操作数的逐个二进制位进行无进位加法,其也称为按位异或.
   ii. 按位加可以用于翻转一个整数的某些位.
   iii. 大小写转换: `char^=32`

移位运算

- 运算规则:将左操作数的每位向左(<<)或向右(>>)移动由右操作数指定的位数.
- 左移时,左边高位被移出,右边空出的低位补0;
- 右移时,右边低位被移出,左边高位可能填充0,也可能填充1,如果左操作数是无符号类型则填充0,有符号则依赖于系统.
- 只要不丢失有效位, `a<<n` 相当于$a*2^n$; `a>>n` 相当于$a/2^n$.

类型转换

类型转换的规则

1. **整数提升**
   任何表达式(除了赋值表达式)中的有符号和无符号的**char和short类型**都被自动转换成**int**或

unsigned,如果原始类型的所有值都能用int表示,则转换成int,否则转换成**unsigned**.

2. **算术转换**
   当双目运算符的操作数求值时,先对每个操作数**独立进行整数提升**,提升后如果两个操作数类型不同,就会发生算术转换.其原则是:值域较窄的向值域较宽的类型转换.
   char/short→int→unsigned→long→unsigned long→long long→unsigned long long→float→double→long double
   E.G.
   可以通过算数转换规则进行运算结果的类型转换,例如 `printf("%.2f%",1.0*sum/(N-2));` ,其中1.0天然为double类型常量.
3. **赋值转换**
   赋值操作中,右操作数的值被完全转换为左操作数的类型.赋值转换不受算数转换规则的约束,赋值表达式结果的类型完全由赋值运算符**左操作数的类型决定**,这可能会导致数据失真(精度降低).
4. **强制类型转换**
   可以利用强制类型转换符,将一个操作数转换成所需类型.其形式为:**(类型名)操作数**.
   E.G.
   `(long)('a'-32)` :先将'a'自动提升成int类型,然后与32相减,最后强制转换为long类型.
   强制类型转换有时可以用作**取整**

类型转换的方法

1. 整数之间转换
   i. 无符号整数
      a. 转换为较短整数时,保留低位**截去多余高位**;
      b. 转换为长度相同的有符号整数时,最高位变成符号位;
      c. 转换成较长整数时,用0填充高位(**0扩展**).
   ii. 有符号整数
      a. 转换为较短整数时,只**保留低位值**
      b. 转换为长度相同的无符号整数时,最高位失去符号功能;
      c. 转换为较长整数时,用符号位填充扩展的高位(**符号扩展**)
         E.G.
         `-1<1U` 解释为0xffffffff<1
2. 整数与浮点数之间转换
   浮点数转换为整数时**截去小数部分**,再将整数部分转换为指定类型.

枚举类型

枚举类型的声明

enum被用于定义枚举类型,它允许对一个几乎命名,并声明集合中包含的标识符及其值,声明形式为:
enum [枚举名] {标识符[=常量表达式],标识符[=常量表达式],...};

- 枚举名是标志该枚举类型的标识符,可以不用枚举名;
- {}是枚举符表,其中逗号隔开的每一个部分称为一个**枚举符**,每个枚举符定义一个用标识符命名的int常量,称为**枚举常量**.;
- 在未指定枚举常量的值的默认情况下,第一个枚举常量的值是0,以后依次递增1;
- 定义时可以指定一个枚举常量的值,未指定的常量比前面的值大1;
- 枚举常量就是整型常量,使用枚举常量是为了让程序更易读.一个枚举类型中不同的标识符可能有相同的值.

枚举类型定义符号常量

#define允许单独为每一种情况命名,但是不能将这些常量组成一组相关项.枚举类型定义通常用于给一组相关联的整型常量命名.

枚举变量的定义

- 在声明枚举类型的同时定义枚举变量;
- 利用枚举名定义枚举变量.
- E.G.
  `enum color {RED,GREEN,BLUE} c1,c2;` ,声明语句声明了一个color枚举类型,同时也定义了该类型的两个变量c1和c2.也可以用enum color定义该类型的其他变量,可以写成下面形式:

```
enum color {RED,GREEN,BLUE};
enum color c1,c2;
```

- 一个枚举变量的值是int型整数,但值域仅限于列举出来的范围(这里是0,1,2);
- 枚举变量的输入与输出都只能是整数,枚举常量可以赋值给同类型的枚举变量,也可以赋给整型变量.
- 还可以通过typedef来定义枚举类型常量和变量:

```
typedef enum {RED,GREEN,BLUE} color;
color c1,c2;
```

- 使用typedef关键字,将这个枚举类型定义为一个新的类型名color.这种方式的优点是,可以将 color 作为一种类型直接使用，而不需要再次声明它是一个枚举类型。