C语言入门（*CS100*）Part One

> *-- By Yiming Li*

> 本文章的内容参考至*Stephen Prata*的《C Primer Plus》

> 本内容适用于*CS*专业学生

int类型认识

c语言中存在三个附属关键字修饰的基本整数类型：

1. short int（或者简写short）类型：占用的储存地方比较小用于较小数值的情况
2. long int(或者简写long)类型：占有的储存地方比较多

long和short都是有符号类型

3. unsingned int或者unsigned只用于表示非负的情况

现在再计算机中规定long long占64位；long占32位；short占16位;int占16位或者是32位

```c
//整数溢出的情况
#include <stdio.h>
int main()
{
    int i = 2147483647;
    unsigned int j = 4294967295,q = 0;

    printf("%d,%d,%d",i,i+1,i+2);
    printf("%u,%u,%u,%u,%u",j,j+1,j+2,q,q-1);
    return 0;
}
```

```
输出结果：
2147483647,-2147483648,-2147483647
4294967295,0,1,0,4294967295
```

- 打印short,long,long long,unsigned的类型：

```c
/*print2.c--更多printf()的特性*/
#include <stdio.h>
int main()
{
    unsigned int un = 3000000000;
    short end = 200;
    long big = 65537;
```

```
    long long verybig = 12345678908642;

    printf("un = %u and not %d\n", un, un);
    printf("end = %hd and %d\n", end, end);
    printf("bog = %ld and not %hd\n", big, big);
    printf("verybig = %lld and not %ld\n", verybig, verybig);
    return 0;
}
```

```
输出结果（可能不同）
un = 3000000000 and not -1294967296
end = 200 and 200
bog = 65537 and not 1
verybig = 12345678908642 and not 1942899938
```

- 程序解释：
    1. 使用错误的转换说明会得到意想不到的结果。第一行输出，对于un(无符号变量)，使用%d会生成负值！其原因是无符号值3000000000和有符号值-1294967296在系统内存中的内部表示完全相同。因此如果告诉printf()函数是一个无符号数，他打印一个值，如果告诉他是有符号数，他打印另外一个值。但是在较小的数字如96就不会出现这样的情况。
    2. 第二行输出，对于short类型的变量end,在printf()中无论是使用(%hd)还是%d都是一样的结果，这是因为在计算机读取short类型的时候都会先将short类型转换成int类型。那为什么要这样多此一举呢？因为在计算机中使用int类型传输数据要更快。
    3. 但是%hd中的h又有什么作用呢？其实在这个地方h可以显示将较大整数截断成short的情况。将65537写成32位的时候位00000000000000001000000000000001(第32位和第17位为1，其余位均为0);使用%hd只会查看后面16位，所以显示的就是1.
    4. 与3.相同，verybig使用%lld查看完整值，使用%ld只是查看后面32位。

注意：

在C程序中一定要注意：直接写出来的计算的模式268413354+15623215是默认在int范围中的计算,会超过int的范围；哪怕在后面赋值给long long int = 268413354+15623215;也会出现溢出的情况：要使用long long int = 268413354LL+15623215LL才能避免这个情况出现。

有符号整数类型总结

1. **char**
    - 范围：-128 到 127（如果char是有符号的）或 0 到 255（如果char是无符号的）
    - 大小：1字节 (8位)
2. **short int 或 short,(%hd)**
    - 范围：-32,768 到 32,767
    - 大小：至少2字节 (16位)
3. **int,(%d/%i)**

- 范围：-32,768 到 32,767（最小要求）
- 大小：至少2字节 (16位)
- 实际上在大多数现代系统中，`int`通常是4字节 (32位)，范围为 -2,147,483,648 到 2,147,483,647

4. **`long int` 或 `long`,(`%ld`)**

   - 范围：-2,147,483,648 到 2,147,483,647（最小要求）
   - 大小：至少4字节 (32位)
   - 在64位系统中，`long`可能是8字节 (64位)，范围为 -9,223,372,036,854,775,808 到 9,223,372,036,854,775,807

5. **`long long int` 或 `long long`,(`%lld`)**

   - 范围：-9,223,372,036,854,775,808 到 9,223,372,036,854,775,807
   - 大小：8字节 (64位)

无符号整数类型总结

1. **`unsigned char`**

   - 范围：0 到 255
   - 大小：1字节 (8位)

2. **`unsigned short int` 或 `unsigned short`,(`%hu`)**

   - 范围：0 到 65,535
   - 大小：至少2字节 (16位)

3. **`unsigned int`,(`%u`)**

   - 范围：0 到 65,535（最小要求）
   - 大小：至少2字节 (16位)
   - 实际上在大多数现代系统中，`unsigned int`通常是4字节 (32位)，范围为 0 到 4,294,967,295

4. **`unsigned long int` 或 `unsigned long`,(`%lu`)**

   - 范围：0 到 4,294,967,295（最小要求）
   - 大小：至少4字节 (32位)
   - 在64位系统中，`unsigned long`可能是8字节 (64位)，范围为 0 到 18,446,744,073,709,551,615

5. **`unsigned long long int` 或 `unsigned long long`,(`%llu`)**

   - 范围：0 到 18,446,744,073,709,551,615
   - 大小：8字节 (64位)

使用字符`char`类型

- `char`类型用于储存字符（如，字母和标点符号），但是从技术层面来讲，`char`是整数类型。因为`char`类型实际上储存的是整数而不是字符。计算机使用数字编码来处理字符，即使用特定的整数来代表字符。（美国使用ASCII编码，本书也是用这个编码，许多IBM的大型主机使用的是另一种编码——EBCDIC，其原理也相同）

- 标准ASCII编码的范围是0~127,只需要7位2进制数即可表示。通常char类型被定义为 8位的储存单元，因此容纳ASCII完全足够。
- C语言吧一字节定义为char类型占用的位（bit）数，因此无论是16位还是32位系统，都可以使用char类型。

声明char类型

char类型的声明方式与其他类型的声明方式相同。下面是一些例子：

```
char respones;
char itable, latan;
```

以上创建了3个char类型的变量:response,itable,latan;

字符常量的初始化：

```
char grade = 'A';
```

在C语言中，用单引号括起来的单个字符被称之为字符常量(character constant)。下面是一些其他的例子：

```
char broiled;
broiled = 'T';
broiled = T; //错误！此时T为一个变量
broiled = "T"; // 错误！此时T为一个字符串
char res = 'FATE';//最终的res = 'E'
```

- 注意ASCII编码使用时，注意数字和字符之间的区别：'4'对应的ASCII编码是52,'4'表示的是字符4而不是数字4.

非打印字符

单引号只适用于字符，数字和标点符号。浏览ASCII就会发现，有一些ASCII字符无法打印出来。例如有以下表示行为的字符(如，退格，换行，终端响铃，蜂鸣)。C语言提供3中方法表示这一些字符。

1. 使用ASCII码：char beep = 7;这里蜂鸣字符对应的ASCII字符是7.
2. 使用转义字符：char nerf = '\n';.

转义序列	含义
\a	警报
\b	退格
\f	换页
\n	换行

转义序列	含义
\r	回车
\t	水平制表符
\v	竖直制表符
\\	反斜杠
\'	单引号
\"	双引号
\?	问号
\0oo	八进制(oo必须是有效的八进制数，即每一个o可以表示0~7中的一个数)
\xhh	十六进制(hh必须是有效的八进制数，即每一个h可以表示0~h中的一个数)

整型常量的例子

类型	十六进制	八进制	十进制
char	\x41	\0101	\65/65
int	0x41	0101	65
unsigned int	0x41u	0101u	65u
long	ox41L	0101L	65L
unsinged long	ox41UL	0101UL	65UL
long long	0x41LL	0101LL	65LL
unsigned long long	0x41ULL	0101ULL	65ULL

打印字符

```c
#include <stdio.h>
int main ()
{
  char ch;
  printf("Please enter a character.\n");
  scanf("%c",&ch);
  printf("The code for %c is %d.\n",ch,ch);
  return 0;
}
```

课件要输出char类型一定要使用%c来完成，如果使用%d输出的就是它对应的ASCII编码。

有符号还是没符号

编译器默认char是无符号

_Bool类型

true和false.C语言用1表示true使用0表示false.所以_Bool也是一种整数类型。但是他也只需要*1bit*的储存空间。

float,double,long double

各种整数类型对于大多是软件开发够用了，但是面向金融和数学的程序经常使用浮点数。C语言中的浮点数类型有float,double类型。

计数法示例

数字	科学计数法	指数计数法
1000000000	1.0×10^9	$1.0e9$
12300	1.23×10^5	$1.23e5$
322.56	3.2256×10^2	$3.2256e2$
0.000056	5.6×10^{-5}	$5.6e-5$

- 在C语言的标准中float可以表示至少6位有效数字，且取值范围可以从10^{-37}到10^{+37}。通常一个计算机使用32位来表示一个float，期中8位表示指数的值和符号，剩下的24位表示非指数的部分。（尾数或是有效数）
- C语言提供的另外一种浮点数的表示方法位double,它的精度更高计算机使用64位来储存它。还有long double它的精度更高。

声明浮点数类型

```
float noah,jobah;
double trouble;
float planck = 6.63e-39;
long double gnp;
```

浮点类型常量

- 在表示的时候需要注意一下几点：
    1. + 可以省略
    2. 可以没有小数点（2E5）或者指数部分，但是不能够两者都没有。
    3. 一些正确的示例：3.14159,.2,4e12,.8E-5,100.;
    4. 在表示小数的时候中间不能够存在空格
- 在C语言中，没有后缀的本系统默认为double类型，可以在后面加上f表示为float类型，或者使用L表示为long double，虽然double类型表示的非常精确，但是运算的速度较慢。

打印浮点数

```c
#include <stdio.h>
int main()
{
    float about = 3200.0;
    double abet = 2.14e9;
    long double dip = 5.32e-9;
    printf("%f can be written %e\n",about,about);
    printf("And it's %a in hexadecimal,powers of 2 notation\n",about);
    printf("%Lf can be written %Le\n",dip,dip);
}
```

- 我们使用%f来表示一般的小数，但是我们使用%e来表示使用指数表示的小数。
- 使用%Lf来表示long double;使用%Le来表示使用指数表示的long double.

浮点值的上溢和下溢

- 这种情况很解释起来很简单，就是计算机无法运算特别大/小的答案

```c
#include <stdio.h>
int main(){
    float too_big = 3.4E38*100.0f;
    printf("%e\n");
    return 0;
}
```

```
输出示例:
inf
```

- 原因解释：这个inf表示的就是无穷大的意思，可见在这里过大了。
- 同样的向下溢出也就是数据过小了。

参数和陷阱

```c
//badcount.c--参数错误的情况
#include <stdio.h>
int main(){
    int n = 4;
    int m = 5;
    float f = 7.0f;
    float g = 8.0f;

    printf("%d\n",n,m);//参数少了
    pinrtf("%d,%d,%d\n",n);//参数多了
    printf("%d %d\n",f,g);//参数不匹配
}
```

- 如果参数少了：会输出相应个数的参数对应的数据。
- 如果参数多了：参数多了，后面的参数就是未定义的情况，会随意输出一个符合参数类型的数据。
- 如果参数不匹配：也会导致未定义的情况：会随意输出一个数值。（可以互相转换的：公用一个输出：double,float;有的时候int,long,short之间也可互相转化）

转义序列示例

```
#include <stdio.h>
int main(){
    float salary;
    printf("\aEnter your desired monthly salary:");
    printf("$_____\b\b\b\b\b\b\b");
    sacnf("%f",&salary);
    pirntf("\n\t$%.2f a month is $%.2f a year",salary,salary*12.0f);
    printf("\rGee!\n");
    return 0;
}
```

字符串和格式化输入/输出

引入例子

```
#include <stdio.h>
#include <string.h> // 提供strlen()函数的原型
#define DENSITY 62.4 //人体密度（单位：磅/立方英尺）
int main()
{
    float weight,volume;
    int size,letters;
    char name[40]; // name是一个可以容纳40个字符的数组
    printf("Hi! What's your first name?\n");
    scanf("%s",name);
    printf("%s What's your weight in pounds?\n",name);
    scanf("%f",&weight);
    size = sizeof(name);
    letters = strlen(name);
    volume = weight/DENSITY;
    printf("Well, %s, your volume is %2.2f cubic feet.\n",name,volume);
    printf("Also, your first name has %d lettes,\n",letters);
    printf("and we have %d bytes to store it.\n",size);
    return 0;
}
```

```
输出结果：
Hi! What's your first name?
LiYiMing
```

```
LiYiMing What's your weight in pounds?
64.5
Well, LiYiMing, your volume is 1.03 cubic feet.
Also, your first name has 8 lettes,
and we have 40 bytes to store it.
```

该程序包含下面新特性

- 用数组（**array**）储存字符串(character string)。在给程序中，用户输入的名字被储存在数组中，该数组重用内存中40个连续的字节，每一个字节储存一个字符值。
- 使用`%s`转换说明来处理字符串的输入和输出。注意，在`scanf()`中`name`没有`&`前缀，而`weight`有（后续解释）
- 用C的预处理将`DENSITY`处理成常量
- 使用`strlen()`来获得字符串的长度

字符串简介

`char`类型和`null`类型

- C语言中没有专门存放字符串的地方，所有的字符串全部储存在`char`类型的数组中。 `this is a string`这样的一句话换算成为`char`的数组的储存的结果是这样的:`this_is_a_string[\0]`(所有的空格都是一位，最后一个`[\0]`是结束的空字符)所有的字符串在储存的时候都存在一个结束空字符`[\0]`所以如果要储存49位的字符串至少要`char name[50]`.
- `char name [50]`这一句话表示的是：`name`是一个数组，`[50]`这一句表示的是总共的储存的位数为50位，`char`表示的意思是每一位内容都是`char`属性的。

> 这样看字符串还挺麻烦的

使用字符串

```c
#include <stdio.h>
#define FRAISE "You are an extraodinary being"
int main()
{
    char name[40];
    printf("What's your name\n");
    scanf("%s",name);
    printf("Hello ,%s.%s\n",name,FRAISE);
    return 0;
}
```

```
What's your name
LiYiming
Hello ,LiYiming.You are an extraodinary being
```

- 注意：如果在输入字符串的时候在中间打了空格就无法储存后面的内容:

```
What's your name
Liyiming 00
Hello ,Liyiming.You are an extraodinary being
```

- %s告诉printf()打印一个字符串。其中一个字符串是储存在name一个是由一开始的PRAISE来定义的。
- 你不用像之前的char name[50]一样考虑结尾的空字符，scanf()函数会自动加上这个空字符。
- 注意scanf()函数在读取在第一个空格（制表符或者是换行符）就不会再继续读取了。C语言还有其他的读取字符串的方法fgets()用于读取一般的字符串
- **字符和字符串的区别**:注意""括起来的叫做字符串;''括起来的叫做字符

strlen函数

- strlen()函数给出字符串中的字符长度(包括其中的字符，空格，换行符等转义字符：\n等转义字符只算一个字符长度)
- 在使用strlen()函数的时候要在前面导入#include <srting.h>库

常量和C预处理器

- 先看一个例子circunference = 3.14159 * diameter这里的3.1415代表著名的$\pi=3.14159\dots$如果可以使用一个常量保存这一个值就可以了
- 为什么不直接在程序中声明这一个变量？Ans:因为在程序前声明变量操作容易而且不容易在程序中被修改。
- 在C语言中存在预处理器；这样的话一开始的内容如#include <stdio.h>
- 与此相同在主程序之前加一个#define NAME value这个就叫做明示常量(*manifest constant*)
- 注意#define NAME value后面不用打;
- #diefine还可以定义字符或者是字符串常量

```
#define BEEP '\a'
#define TEE "T"
#define ESC '\033'
#define OOPS "Now you have done it!"
// 下面是一些错误的示范
#define TOES = 20 // 这里的TOES等于'= 20'
// 如果这么写:
digits = fingers + TOES;
// 结果就是 digits = fingers += 20
```

- 使用#define定义的常量在程序中如果尝试进行修改这样的话会导致编译错误：error: assignment to expression with array type

const限定符

- 同样和#define一样的还有const这个函数,它也可以定义常量，同时也是只读不能修改
- const char FRAISE[] = "You are a wonderful person!!!"; 在这个地方有几个地方和#define不同：
    1. #define后面不需要使用;

2. `#define`后面直接写大写的常量名称和赋值内容；但是`const`后面要写明是什么类型然后再是所有的内容。

- C语言的`<limits.h>`和`<float.h>`分别提供了整数类型和浮点类型大小限制相关的详细信息。每个头文件都定义类一系列供实现使用的明示常量。例如`<limits.h>`头文件包含下列类似的代码:

```
#define INT_MAX +32767
#define INT_MIN -32767
```

- 这样的话如果有`#include <limits.h>`这个头文件就可以编译`printf("Maximum int value on this system = %d\n",INT_MAX);`
- 具体的`<limits.h>`和`<floats.h>`这些头文件包含的内容见书*P68*

printf()和scanf()

输出

- `pirntf()`中怎么打印`%`和`\`?
- `printf("%%d = %d \\ \n",20);`输出结果`%d = 20 \`
- 在`%`和转换字符之间加入数字可以实现基本的转换说明，例如`%4d`就表示的是字符宽度就是4；还有`%5.2f`就是表示总共宽度是5，保留小数点后两位。
- `%+6.2f`就表示在如果这个数值为正就在前面加上`+`,如果是负数就在前面加上`-`

输入

- `scanf()`其实就是把输入的字符串转化成为整数，浮点数，字符，字符串，而`printf()`正好与他相反，将整数，浮点数，字符和字符串都转化成为屏幕上的文字。

```
#include <stdio.h>
int main()
{
    int age;
    float asstes;
    char pet[30];

    printf("Enter your age,assets, nad favorite pet.\n");
    scanf("%d %f",&age,&assets);
    scanf("%s",pet); // 字符数组不需要使用&
    printf("%d %s %.2f\n",age,pet,assets);
    return 0;
}
```

输出示例:

```
Enter your age,assets, nad favorite pet.
12 13.568 dog
12 dog 13.57
```

```
Enter your age,assets, nad favorite pet.
12
13.568
dog
12 dog 13.57
```

- 解释在输入的时候 和\n都算作是下一个内容的读取这也是为什么在之前我们输入字符串的时候不能够读取空格，换行符之后的内容一样
- 同样的在这个地方也可以将两个scanf()的内容合并：scanf("%d %f %s",&age,&assets,pet);
- 什么时候需要&在scanf()函数中?
    - 当基本数据类型int,float,double中需要使用&传递变量的地址
    - 对于字符数组而言（字符串之类），直接传递字符数组的名称即可

格式字符串中的普通字符

- 使用scanf("%d%f%s",age,assets,dog);这个句子的时候其实每一个数据类型**前面(注意不是数据的后面)**的空格scanf()函数都不会录入进去所以使用scanf("%d %f %s",age,assets,dog);和scanf("%d%f%s",age,assets,pet);这两句的效果是一样的。
- 但是也可以使用不同的分割效果scanf("%d,%f,%s···")这样的句子表示每一个数据之间的分割就是由,来进行分割的。