3.5 综合应用

郭延文

2019级 计算机科学与技术系

循环的重要性

- 完成重复操作任务的程序，如求阶乘、求泰勒展式...
  - 100! = 1*2*3*...*100;
  - n!; // n可能在程序运行期间才能确定
- 完成遍历性操作
  - 从1号同学到179号同学 ...
  - 对数组元素进行操作
  - 对链表操作
- “综合”流程控制
  - 综合分支流程控制方法
  - 综合用到顺序、循环(for/while)和选择(if…else../switch)
  - 可能用到break/continue/goto…

循环流程控制方法的综合运用
例：求斐波那契级数

- 例:（斐波那契Fibonacci数列）有一对兔子，从出生后第3个月起每个月都生一对兔子，小兔子长到第3个月每个月又生一对兔子，假设所有兔子都长生不老，求第n个月的兔子总对数。

1，1，2，3，5，8，13，…

Fibonacci 数的定义如下：

$$fib(n) = \begin{cases} 1 & (n=1) \\ 1 & (n=2) \\ fib(n-2) + fib(n-1) & (n \geq 3) \end{cases}$$

要想求第n项的值，必须先依次计算前面n-1项的值。
可以运用循环流程实现这类“依次”求解问题。

```
…
int main()
{       int n, num = 0;
        scanf("%d", &n);
        int fib_1 = 1, fib_2 = 1;
        for (int i = 3; i <= n; i++)
        {
                num = fib_1 + fib_2;
                //计算第i项
                fib_1 = fib_2;
                //第i-1项为下一个i的第i-2项
                fib_2 = num;
                //第i项为下一个i的第i-1项
        }
        printf("第 %d 个月有 %d 对兔子.\n", n, num);
        return 0;
}
```

```
…
int main()
{       int n, num = 0;
        scanf("%d", &n);
        int fib_1 = 1, fib_2 = 1;
        for (int i = 3; i <= n; i++)
        {
                num = fib_1 +
                //计算第i项
                fib_1 = fib_2;
                //第i-1项为下
                fib_2 = nu
                //第i项为下一
        }
        printf("第 %d 个月有 %d 对兔子.\n", n, num);
        return 0;
}
```

后次循环的计算依赖于前次循环，循环间有强耦合性，难以用OpenMP“简单”加速

以时间换空间！

```
fib_2 = fib_1 + fib_2;
//计算第i项，并作为下一个i
// 的第i-1项
fib_1 = fib_2 - fib_1;
//第i-1项作为下一个i的第i-2项
```

思考题：

- 给定两个数a 和 b

  问：不借助第3个变量，如何实现这两个数的交换？

例：求所有的三位水仙花数

- 设计程序，求所有的三位水仙花数，一个三位水仙花数等于其各位数字的立方和，比如：

$$153 = 1^3 + 3^3 + 5^3$$

- 分析：
  - 对于一个三位数，其百位数字可能为1,...,9，其十位与个位数字可能为0,1,...,9，假设分别用i,j,k表示，则这个三位数为i*100 + j*10 + k。
  - 根据题意，可以通过依次改变i,j,k，列举出所有的三位数，将符合条件的数输出。
  - 这是一种列举（又叫枚举）的方法，可以运用循环流程实现其中的“依次”列举。

以上列举的过程通过嵌套循环实现

```
…
   for (int i = 1; i <= 9; i++)
   {
         for (int j = 0; j <= 9; j++)
         {
               for (int k = 0; k <= 9; k++)
                     if (i*100 + j*10 + k == i*i*i + j*j*j + k*k*k)
                           printf("%d \n", i*100 + j*10 + k);
               // 存在一定的重复计算！可以以空间换时间
         }
   }
   …
```

```
…
    for (int i = 1; i <= 9; i++)
    {
        int n0 = i * 100, sum0 = i * i * i;      // 以空间换时间
        for (int j = 0; j <= 9; j++)
        {
            int n1 = n0 + j * 10, sum1 = sum0 + j * j * j;
            for (int k = 0; k <= 9; k++)
                if (n1 + k == sum1 + k * k * k)
                    printf("%d \n", n1 + k);
        }
    }
    …
```

再回首：例：求所有的三位水仙花数

遍历N=100-999?

给定一个正整数i，如何分离出他的各个位上的数字?

借助/,%

这两个运算

例 统计交通流量


法制晚报


收费站
ETC专用
ETC专用
非ETC车辆禁止通行
非ETC车辆禁止通行
20
20
ETC专用
保持车距

例 统计交通流量

路边设置一车辆探测器，探测器用线路连接到计算机，

(a) 当有车辆通过时，探测器传送信号1给计算机，

(b) 探测器中有一计时器，每秒钟发出一个数字信号2传给计算机，

(c) 探测结束时传递一个数字信号0给计算机

假设一次探测从开始到结束，共发出了类似以下的一组数字给计算机：

1 2 1 1 2 2 1 2 1 2 0

编写程序读入这一系列的信号并输出：

① 进行了多长时间的统计调查？

② 记录到的车辆数？

③ 在车辆之间最长的时间间隔是多少？

5秒内，有5辆车通过，
最长隔2秒有车通过

问题分析：

1 2 1 1 2 2 1 2 1 2 0

输入信号(sign)
- 1(车辆通过)
- 2(秒钟信号)
- 0(探测结束)

输出结果
- ①进行统计的时间(seconds)
- ②记录到的车辆数(nums)
- ③在车辆之间最长的时间间隔(longest)
  由此派生出的车辆间隔变量(inter)

根据以上数据特性，可确定数据类型为整型(int)

算法设计：（自顶向下，逐步求精）

1 2 1 1 2 2 1 2 1 2 0

一级流程

```
begin
        1.数据定义及初始化
        2.读入探测信号sign
        3.while( sign != 0 )
                对sign进行处理
        4.输出结果
end
```

二级流程

```
while( sign != 0 )
{   if ( sign == 1 )
      3.1处理车辆信号;
    else if ( sign == 2 )
      3.2处理计时信号;

    3.3读入下一个sign;
}
```

三级流程

```
begin
      3.1.1 车辆计数nums++;
      3.1.2 处理最大间隔
        if(longest<inter)
                longest=inter;
      3.1.3 为下一个间隔计数作准备
         inter=0;
end
```

```
begin
      3.2.1 总时间累加器seconds++
      3.2.2 间隔计数器inter++;
end
```

1 2 1 1 2 2 1 2 1 2 0

```
    int sign;
    int nums=0, seconds=0, inter=0, longest=0;
// num 车辆数; seconds 时间; inter 车辆间隔; longest 最大间隔
    scanf("%d", &sign);
    while(sign != 0)
    {       if (sign == 1)
            {
                nums++;
                if (inter > longest)    longest = inter;
                inter = 0;
            }
           else if (sign == 2)
           {
                seconds++;
                inter++;
           }
          scanf("%d", &sign);   // C++ scanf -> cin, printf -> cout
    }
    printf("nums=%d, seconds=%d, longest=%d.\n", nums, seconds, longest);
    …
```

Input sign:

1 2 1 1 1 1 2 1 1 1 2 1 1 1 1 2 1 1 2 1 1 2 1 2 0

nums=17, seconds=7, longest=1.

例：从键盘接收字符，一直到输入了字符y(Y)或n(N)为止。（事件控制的循环 ）

例：从键盘接收字符，一直到输入了字符y(Y)或n(N)为止。（事件控制的循环 ）

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{    char ch;
     do
     {    cout << "请输入Yes或No（y/n）：";
          cin >> ch;
          ch = tolower(ch);
     } while (ch != 'y' && ch != 'n');
     if (ch == 'y')
          ......
     else
          ......
     return 0;
}
```

例：编程求出小于n的所有素数（质数）

```
#include <iostream>
using namespace std;
int main()
{    int n;
     cout <<"请输入一个正整数："
     cin >> n; //从键盘输入一个正整数
     for (int i=2; i<n; i++) //循环：分别判断2、3、…、n-1是否为素数
     {     int j=2;
           while (j < i && i%j != 0) //循环：分别判断i是否能被2～i-1整除
                j++;
           if (j == i) //i是素数
                cout << i << " ";
     }
     cout << endl;
     return 0;
}
```

注意：1、上面的for循环中，偶数没有必要再判断它们是否为素数；

2、上面的while循环没有必要到i-1，只需要到：sqrt(i)

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{    int n;
     cin >> n;  //从键盘输入一个数
     if (n <= 2) return -1;
     cout << 2 << ","; //输出第一个素数
     for (int i=3; i<n; i+=2) //循环：分别判断3、5、…、是否为素数
     {    int j=2;
          while (j<=sqrt(i) && i%j!=0) //循环：分别判断i是否为素数
               j++;
          if (j > sqrt(i)) //i是素数
               cout << i << ",";
     }
     cout << endl;
     return 0;
}
```

注意：上面程序中的sqrt(i)被重复计算！

```
……
int j = 2, k=sqrt(i);
while ( j <= k && i%j != 0)
      j++;
if (j > k) //i是素数。
……
```

注意：对有些循环优化（循环之间耦合性不强！），编译器能实现！

Intel OpenMP

例：用牛顿迭代法求 $\sqrt[3]{a}$

- 计算 $\sqrt[3]{a}$ 的牛顿迭代公式为：

$$x_{n+1} = \frac{1}{3}(2x_n + \frac{a}{x_n^2})$$

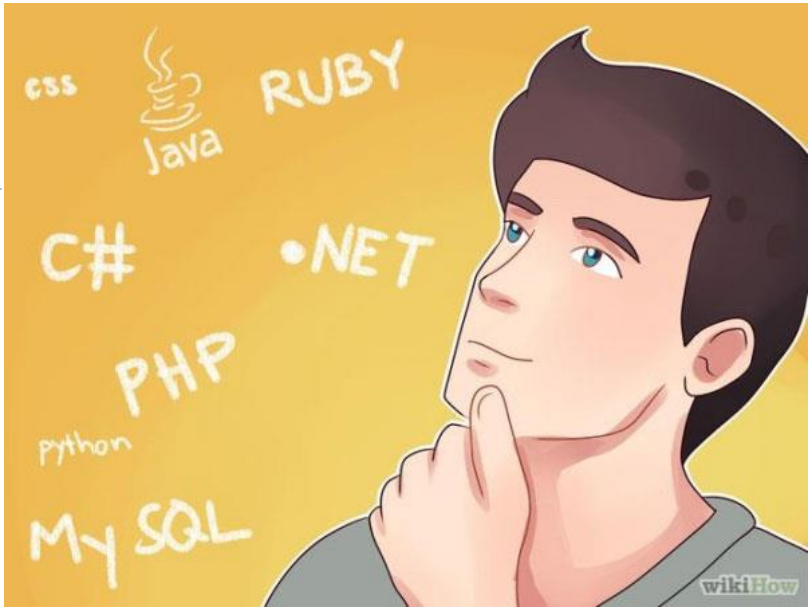
- 取x_0为a（任何值都可以），依次计算x_1、x_2、…，直到：

$|x_{n+1}-x_n| < \varepsilon$（$\varepsilon$为一个很小的数，可设为$10^{-6}$）时为止，$x_{n+1}$即为所求值。

$$x_{n+1} = \frac{1}{3}(2x_n + \frac{a}{x_n^2})$$

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{   const double eps=1e-6;   //一个很小的数
    double a,x1,x2;  //x1和x2分别用于存储最新算出的两个值
    cout << "请输入一个数：";
    cin >> a;
    x2 = a;  //第一个值取a
    do
    { x1 = x2;  //记住前一个值
      x2 = (2*x1+a/(x1*x1))/3;  //计算新的值
    } while (fabs(x2-x1) >= eps);
   cout << a << "的立方根是：" << x2 << endl;
   return 0;
}
```

如何写程序？



分析具体
问题的解法

需要定义什么
变量、
分别什么类型

顺序、分支、
循环流程

将变量整合为
表达式

- 程序每写几行就编译一次（F7）
- 写完一个模块要测试(Ctrl+F5)、要多用几个测试用例
- 运行不正确用断点调试(F5, F10, F11…)

例：计算级数和的值

- 编写程序计算当x=0.5时下述级数和的近似值，使其误差（前后两次级数和的差）小于某一指定的值ZERO(例如：ZERO=0.000001)

$$x - (x^3)/(3*1!) + (x^5)/(5*2!) - (x^7)/(7*3!) + \cdots$$

即：

$$x-\frac{x^3}{3*1!}+\frac{x^5}{5*2!}-\frac{x^7}{7*3!}+\cdots\cdots$$

PS: 自己可以试验不同的精度，看循环次数和收敛情况

上机实验题!

例：求车牌号

- 一辆卡车违反交通规则，撞人后逃逸。现场有三人目击事件，但都没有记住车牌号，只记下车牌号的一些特征。甲说：车牌号的前两位数字是相同的；乙说：车牌号的后两位数字是相同的，但与前两位不同；丙说：四位的车牌号刚好是一个整数的平方。请编写C/C++程序，根据以上线索求出车牌号。

要求：在main()函数中调用cout输出该车牌号；

上机实验题!

小结：C语言的流程控制语句

- 对应基本流程，C语言提供了控制语句
  - if、if-else、switch
  - while、do-while、for
  - C语言里的其他语句（可作为基本流程的子语句）
    - 复合语句
    - 表达式语句
    - 空语句
- 此外，C语言还提供了辅助控制语句
  - break
  - continue
  - goto
  - return

- break与continue辅助循环流程控制时的区别：
  - 执行break后，不再进行条件判断，直接结束流程；
  - 执行continue后，接着进行下一次循环.

小结

- 要求：
  - 会运用基本流程控制语句实现简单的计算任务

    在main函数中完成数据定义、输入、循环处理、输出
  - 能够定位出错行，修改程序中的语法错误
  - 继续保持良好的编程习惯

前几次上机典型问题解析

上机问题（1）

- 创建VS项目控制台程序的步骤一定要熟练！

- 一个项目里面只能有一个main函数，很多同学在一个项目里写完一个程序后，又添加了另一个程序，这样出现一个项目里有两个或者多个main函数的情况，造成编译出错。解决方法有两个：（1）新建一个项目，在新的项目里写新的程序；（2）将前面的main函数改名或者注释掉，只保留当前需要运行的main程序。

- endl不能出现在cin中： cin>>a>>endl； // 报错。 endl后面的是字母“l”，不是数字“1”，经常有同学搞错！

- 取消输出格式，可以使用resetiosflags()或者流成员函数unsetf()。例如取消fixed的输出格式可以用语句：

  cout<<resetiosflags(ios::fixed)； 或 cout.unsetf(ios::fixed)

- for循环里是“；”，不是“，”。例如：for (int i = 1; i <= 100; i++)，错误是for (int i = 1，i <= 100，i++)

上机问题（2）

- 控制语句的作用范围只能是它之后的一条语句（以分号结束）或者一个{}内的语句。

例（1）

```
int a = 0;
while( a < 100 );
{
    a++;
}
```

上面的程序段会陷入死循环，因为while语句的作用范围会是它后面的分号（一条空语句），这样a的值一直没有变化，程序一直执行这条语句。

例（2）

```
if( a < 3 )
   cout<<9<<endl;
else if( a < 10 )
   double p = a*5;
   cout<<p<<endl;
else
   cout<<a*10<<endl;
```

上面的程序段编译不通过，会提示else找不到if匹配。因为第二条if语句的控制范围只有“double p = a*5;”这条语句，下面的“cout<<p<<endl;”不受其控制，这意味着判断已经结束，因此下面的else找不到与之匹配的if语句，出现编译错误。解决方法就是用大括号将两条语句括起来，使其成为一个受if语句控制的整体。

上机问题（3）

- 变量的定义与初始化：

```
{     int i,j=1; //i和j均已定义,但只有j进行了初始化;
      i+=i; //!!!错误,i未初始化,不能进行操作.
      j+=j; //正确,运行完:j为2
        ...
}
```

- 变量类型及类型转换.计算结果含小数时,要留心精度丢失.

```
int  i=1;
double j;
j=i/3;
//j为0
```

```
double i=1,j;
j=i/3;
//j为0.333333
```

- if和else的作用范围:当if内执行多条语句时,要用{}包住,养成良好的编程习惯.

```
if(条件)
   执行语句1;
   执行语句2;
else ...
```

```
if(条件)
{
    执行语句1;
    执行语句2;
}
else ...
```

上机问题（4）

- 写代码的时候如非必要，关闭中文输入法（“error: stray '\357' in program”通常是中文输入法导致的）
- 有多个条件时用逻辑符号（&&，||）连接，如：x大于15小于等于30应写成（x>15&&x<=30）不能写成（15<x<=30）
- 对于包含字母的代数方程式不要和数学上混淆，例如：5*a+3*b+c/3==100有同学会写成5a+3b+c/3=100
- C++里没有专门的平方运算符。可以调用函数pow(x, n)，其中n表示次数。有同学在写平方时会犯例如“a^2”这样的错误，“^”是位运作符。
- 单独的一条语句：int(y)；并不能直接把y从浮点数变成整数，如：

例：

```
    double y = 1.5;
    int x = int(y);
    cout << "x = " << x << ", y = " << y << endl;
// 输出结果为：x = 1, y=1.5
// y本身不会发生变化，但将转换后的结果赋值给了x。
```

上机问题（5）

- 关于0.0000001的表示:1e-6 或 #define E 0.0000001

  错误表达:10^-6

- ”=”与”==”的区别:虽然基础,但还是很多同学易犯错.”=”为向右赋值运算符,”==”判断左右语句是否相等

- 下次OJ中提交代码前,一定看清当前模板是c还是C++,c模板下提交c++代码是不通过的.

- 代码一定要保证自己敲,复制粘贴易出现编码差异会编译错误,而且这样也不能很好的锻炼自己的编程能力.

一种不提倡的编程语句写法
用break语句协助改写for语句

```
int sum = 0;
for ( int i = 1; i <= 100; i++ )
        sum += i;
```

等价于：

```
int sum = 0;
int i = 1;
for  ( ; ; ) //这种写法虽然编译不出错，但不提倡
{       if(i <= 100)
        {       sum += i;
                i++;
        }
        else
                break;
} // ☹
```

关于程序“排版”– 缩进(tab键)

```
…
    for (int i = 1; i <= 9; i++)
    {
        int n0 = i * 100, sum0 = i * i * i;
        for (int j = 0; j <= 9; j++)
        {
            int n1 = n0 + j * 10, sum1 = sum0 + j * j * j;
            for (int k = 0; k <= 9; k++)
                if (n1 + k == sum1 + k * k * k)
                    printf("%d \n", n1 + k);
        }
    }
```

关于程序命名：工程名、函数名、变量名…

关于字符，字符串的操作

```
char ch;
(ch >= 'a' && ch <= 'z' ) || ( ch >= 'A' && ch <= 'Z')
(ch >= '0' && ch <= '9')
```

字符型数据

- 字符变量：ch
- 字符常量：

  'a' 'z' 'A' 'Z' '0' '9' '\n'

  ASCII字符集：列出所有可用的字符

  每个字符：惟一的次序值（ASCII码

  '0'-'9'

  'A'-'Z'

  'a'-'z'

ASCII码表

符号	10进制	符号	10进制	符号	10进制	符号	10进制
@	64	P	80	`	96	p	112
A	65	Q	81	a	97	q	113
B	66	R	82	b	98	r	114
C	67	S	83	c	99	s	115
D	68	T	84	d	100	t	116
E	69	U	85	e	101	u	117
F	70	V	86	f	102	v	118
G	71	W	87	g	103	w	119
H	72	X	88	h	104	x	120
I	73	Y	89	i	105	y	121
J	74	Z	90	j	106	z	122
K	75	[	91	k	107	{	123
L	76	\	92	l	108	|	124
M	77	]	93	m	109	}	125
N	78	^	94	n	110	~	126
O	79	_	95	o	111	⌂	127

关于字符，字符串的操作

字符型数据的输入和输出

- 字符输入函数getchar

  输入一个字符

  char ch;

  ch = getchar( );

- 字符输出函数putchar

  输出一个字符

  putchar(输出参数);

  字符常量或字符变量

```
char ch;
ch = getchar();
putchar (ch);
putchar ('?');
```

```
a
a?
```

调用scanf和printf输入输出字符

```
double value1, value2;
char operator;
printf("Type in an expression: ");
scanf("%lf%c%lf", &value1, &operator, &value2);
printf("%.2f %c %.2f", value1, operator, value2);
```

关于字符，字符串的操作

字符串：一个特殊的一维字符数组 '\0'

- 把字符串放入一维字符数组（存储）

  数组长度足够

  - 字符数组初始化： char s[20] = "Happy";
  - 赋值： s[0] = 'a'; s[1] = '\0';

    char str[10] = {'H', 'e', 'l', 'l', 'o', ' ', 'N', 'J', 'U', '\0'};

    char str[10] = "Hello NJU";

  - 输入： 输入结束符 ==> 字符串结束符'\0'

```
i = 0;
while((s[i]=getchar( )) != '\n')
    i++;
s[i] = '\0';
```

输出：

```
for(i = 0; s[i] != '\0'; i++)
    putchar(s[i]);
```

关于字符，字符串的操作

- **gets()函数**

用来从标准输入设备(键盘)读取字符串直到回车结束，但回车符不属于这个字符串，其调用格式为:gets(s);

与scanf("%s", &s)相似但不完全相同．使用scanf("%s", &s) 函数输入字符串时如果输入了空格会认为输入字符串结束，空格后的字符将作为下一个输入项处理．但gets() 函数将接收输入的整个字符串直到回车为止。

- **puts()函数**

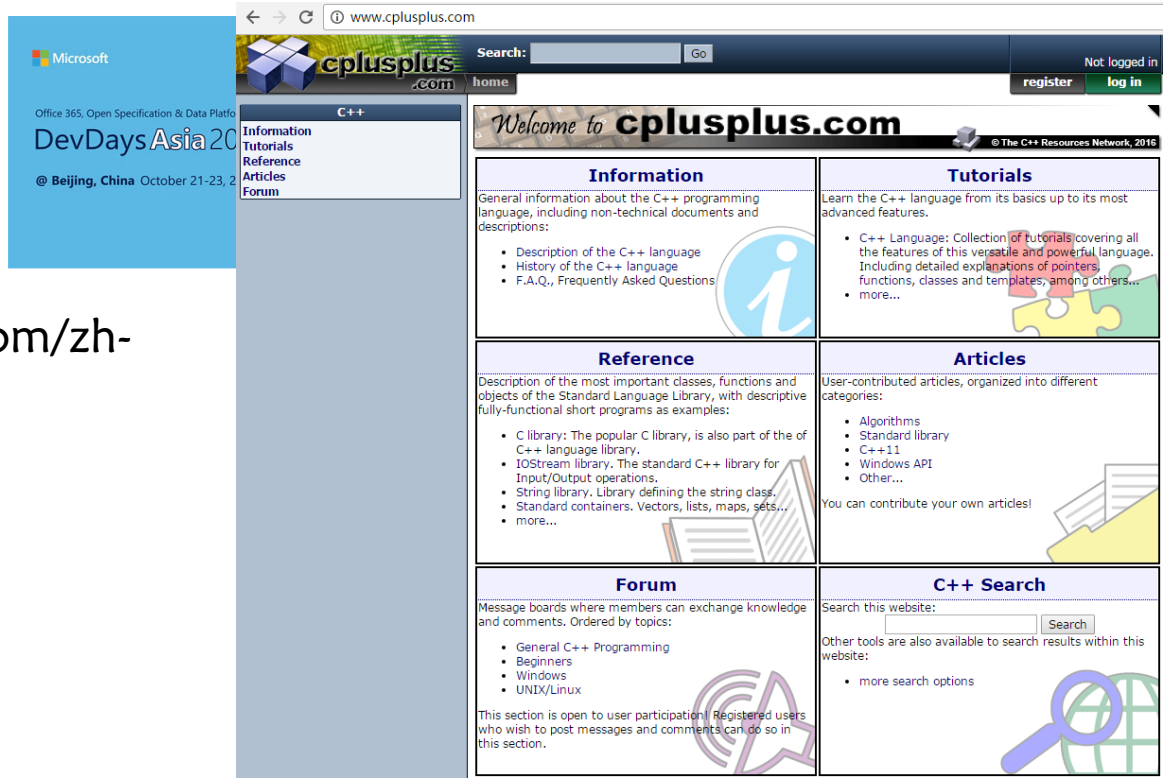
用来向标准输出设备(屏幕)写字符串并换行，其调用格式为:puts(s); 其中s为字符串变量(字符串数组名或字符串指针)。puts()函数的作用与语printf("%s\n",s)相同。

如何查阅函数的使用

DevDays Asia 2016

微软 DevDays Asia 2016 北京技术峰会 & Office 365 应用开发马拉松

MSDN
https://msdn.microsoft.com/zh-cn/default.aspx



http://www.cplusplus.com/

printf、_printf_l、wprintf、_wprintf_l

Visual Studio 2015 其他版本

将格式化输出打印至标准输出流。 有关这些函数的更多安全版本，请参见 printf_s、_printf_s_l、wprintf_s、_wprintf_s_l。

语法

```
int printf(
   const char *format [,
   argument]...
);
int _printf_l(
   const char *format,
   locale_t locale [,
   argument]...
);
int wprintf(
   const wchar_t *format [,
   argument]...
);
int _wprintf_l(
   const wchar_t *format,
   locale_t locale [,
   argument]...
);
```

参数

format
格式控件。

argument
可选参数。

locale
要使用的区域设置。

返回值

返回打印的字符数，或在发生错误时返回负值。 如果 *format* 是 **NULL**，则会调用无效参数处理程序，如 参数验证 中所述。 如果允许执行继续，则该函数返回 -1 并将 **errno** 设置为 **EINVAL**。 如果在 *argument* 中遇到 **EOF** (0xFFFF)，则函数返回 -1。

C++官网

www.cplusplus.com/reference/cmath/pow/?kw=pow

If the *base* is finite negative and the *exponent* is finite but not an integer value, it causes a *domain error*.
If both *base* and *exponent* are zero, it may also cause a *domain error* on certain implementations.
If *base* is zero and *exponent* is negative, it may cause a *domain error* or a *pole error* (or none, depending on the library implementation).
The function may also cause a *range error* if the result is too great or too small to be represented by a value of the return type.

If a *domain error* occurs, the global variable `errno` is set to `EDOM`.
If a *pole* or *range error* occurs, the global variable `errno` is set `ERANGE`.

If a *domain error* occurs:
- And `math_errhandling` has `MATH_ERRNO` set: the global variable `errno` is set to `EDOM`.
- And `math_errhandling` has `MATH_ERREXCEPT` set: `FE_INVALID` is raised.

If a *pole error* occurs:
- And `math_errhandling` has `MATH_ERRNO` set: the global variable `errno` is set to `ERANGE`.
- And `math_errhandling` has `MATH_ERREXCEPT` set: `FE_DIVBYZERO` is raised.

If a *range error* occurs:
- And `math_errhandling` has `MATH_ERRNO` set: the global variable `errno` is set to `ERANGE`.
- And `math_errhandling` has `MATH_ERREXCEPT` set: either `FE_OVERFLOW` or `FE_UNDERFLOW` is raised.

Example

```
1  /* pow example */
2  #include <stdio.h>      /* printf */
3  #include <math.h>       /* pow */
4
5  int main ()
6  {
7    printf ("7 ^ 3 = %f\n", pow (7.0, 3.0) );
8    printf ("4.73 ^ 12 = %f\n", pow (4.73, 12.0) );
9    printf ("32.01 ^ 1.54 = %f\n", pow (32.01, 1.54) );
10   return 0;
11 }
```

Output:

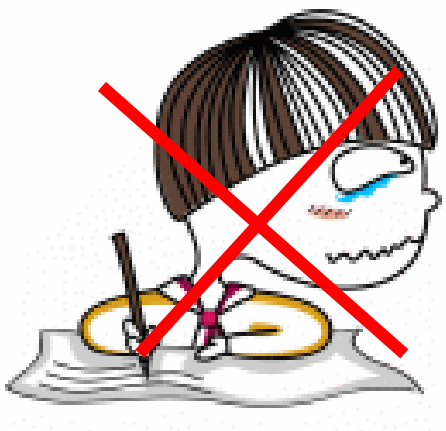
```
7 ^ 3 = 343.000000
4.73 ^ 12 = 125410439.217423
32.01 ^ 1.54 = 208.036691
```

See also

log	Compute natural logarithm (function)
exp	Compute exponential function (function)
sqrt	Compute square root (function)

关于作业的提交

- Deadline: 每周六晚上10点(周六中午12点公布测试用例)
  - 务必独立完成!
  - 每次均会进行查重
  - 根据测试用例验证程序是否正确



- 参考答案程序：

  周六截止后上传至课程网站

Q & A