C++ 程序设计 I

徐东

xu.dong.sh@outlook.com

信息与计算科学

数学系

上海师范大学

2018 年 8 月 26 日

内容

问题

- 编程计算

$$\sqrt{a} \ , \ a > 0$$

问题

- 编程计算

$$\sqrt{a} \quad , \quad a > 0$$

- 迭代公式

$$x_{n+1} = \frac{x_n + \frac{a}{x_n}}{2} \quad , \quad n = 0, 1, 2, \cdots$$

问题

- 编程计算

$$\sqrt{a} \quad , \quad a > 0$$

- 迭代公式

$$x_{n+1} = \frac{x_n + \frac{a}{x_n}}{2} \quad , \quad n = 0, 1, 2, \cdots$$

- $x_0 = \frac{a}{2}$

问题

- 编程计算

$$\sqrt{a} \quad , \quad a > 0$$

- 迭代公式

$$x_{n+1} = \frac{x_n + \frac{a}{x_n}}{2} \quad , \quad n = 0, 1, 2, \cdots$$

- $x_0 = \frac{a}{2}$
- $|x_{n+1} - x_n| \leq 10^{-6}$ 结束迭代

代码

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5     int a = 0;
6     cout << ”input␣a␣:␣”;
7     cin >> a;
8
9     double x0 = a/2, x1 = 0;
10
11    while(true){
12       x1 = (x0 + a/x0) / 2;
13       if((x1 − x0)*(x1 − x0) <= 10.0e−12) break;
14       x0 = x1;
15    }
16
17    cout << ”the␣sqrt␣of␣” << a << ”␣is␣” << x1 << endl;
18
19 return 0;
20 }
```

计算算术平方根

```
1    double x0 = a/2, x1 = 0;
2    while(true){
3        x1 = (x0 + a/x0) / 2;
4        if((x1 - x0)*(x1 - x0) <= 10.0e-12) break;
5        x0 = x1;
6    }
```

- 更简单、安全的方式

计算算术平方根

```
1    double x0 = a/2, x1 = 0;
2    while(true){
3        x1 = (x0 + a/x0) / 2;
4        if((x1 - x0)*(x1 - x0) <= 10.0e-12) break;
5        x0 = x1;
6    }
```

- 更简单、安全的方式
  - $sqrt(a)$

计算算术平方根

```
1    double x0 = a/2, x1 = 0;
2    while(true){
3        x1 = (x0 + a/x0) / 2;
4        if((x1 - x0)*(x1 - x0) <= 10.0e-12) break;
5        x0 = x1;
6    }
```

- 更简单、安全的方式
  - $sqrt(a)$
  - $C++$ 函数

计算算术平方根

```
1    double x0 = a/2, x1 = 0;
2    while(true){
3        x1 = (x0 + a/x0) / 2;
4        if((x1 - x0)*(x1 - x0) <= 10.0e-12) break;
5        x0 = x1;
6    }
```

- 更简单、安全的方式
  - $sqrt(a)$
  - $C++$ 函数
- 函数是用于执行一个特定任务的一组语句（集合）

计算算术平方根

```
1  int a = 0;
2  cout << ”input␣a␣:␣”;
3  cin >> a;
4
5  double x0 = a/2, x1 = 0;
6  while(true){
7     x1 = (x0 + a/x0) / 2;
8     if((x1 - x0)*(x1 - x0)
9     <= 10.0e-12) break;
10    x0 = x1;
11 }
12 cout << ”the␣sqrt␣of␣” << a << ”␣is
       ␣” << x1 << endl;
```

```
int a = 0;                          1
cout << ”input␣a␣:␣”;               2
cin >> a;                           3
                                    4
cout << ”the␣sqrt␣of␣”              5
     << a << ”␣is␣”                 6
     << sqrt(a)                     7
     << endl;                       8
```

cmath

- $<cmath>$ 提供了多个常用的数学函数
  - 指数函数
  - 三角函数
  - 取整
  - ...

cmath

- $< cmath >$ 提供了多个常用的数学函数
  - 指数函数
  - 三角函数
  - 取整
  - $\cdots$
- 在源文件中（顶部），添加

$$\#include < cmath >$$

计算算术平方根

```
#include <iostream>
#include <cmath>
using namespace std;

int main(){
    int a = 0;
    cout << "input a : ";
    cin >> a;
    cout << "the sqrt of " << a << " is " << sqrt(a) << endl;

    return 0;
}
```

指数函数

函数	说明
exp(x)	返回 e 的 x 次方
log(x)	返回 x 的自然对数
log2(x)	返回 x 的以 2 为底的对数
log10(x)	返回 x 的以 10 为底的对数
pow(a, b)	返回 a 的 b 次方
sqrt(x)	返回 x 的算术平方根 (x≥0)
cbrt(x)	返回 x 的立方根 (≥ C++ 11)

三角函数

函数	说明
sin(radians)	返回以弧度为单位的角的正弦
cos(radians)	返回以弧度为单位的角的余弦
tan(radians)	返回以弧度为单位的角的正切
asin(x)	返回以弧度为单位的反三角正弦值
acos(x)	返回以弧度为单位的反三角余弦值
atan(x)	返回以弧度为单位的反三角正切值

- 注意：弧度制!!!

三角函数的使用

```
1   #include <iostream>
2   #include <cmath>
3   using namespace std;
4
5   const double PI = 3.14159265; //全局常量
6
7   int main()
8   {
9       double param = 30.0;
10
11      cout << param << "度角的正弦 = "
12           << sin(param * PI / 180) << endl;
13
14      cout << "正弦值为 " << sin(param * PI / 180) << " 的角度（弧度） = "
15           << asin(sin(param * PI / 180)) << endl;
16
17      return 0;
18  }
```

取整函数等

函数	说明
`ceil(x)`	返回 x 向上取整为离它最近的整数
`floor(x)`	返回 x 向下取整为离它最近的整数
`fmax(x, y)`	返回 x 和 y 中的最大值 (≥ C++ 11)
`fmin(x, y)`	返回 x 和 y 中的最小值 (≥ C++ 11)
`fabs(x)`	返回 x 的绝对值
`abs(x)`	返回 x 的绝对值

取整等函数的使用 I

```
1  #include<iostream>
2  #include<cmath>
3  using namespace std;
4
5  int main()
6  {
7      cout << ceil(2.1) << endl;
8      cout << ceil(2.0) << endl;
9      cout << ceil(-2.1) << endl;
10
11     cout << floor(2.1) << endl;
```

取整等函数的使用 II

```
12      cout << floor(2.0) << endl;
13      cout << floor(-2.1) << endl;
14
15      cout << fmax(2, 3) << endl;
16      cout << fmin(2.5, 4.6) << endl;
17
18      cout << fabs(-2.1) << endl;
19      cout << abs(-2.1) << endl;
20
21      return 0;
22  }
```

问题

- 输入三角形的三个顶点，计算并显示三个角的角度。
- 角度计算公式

$$A = acos(\frac{a^2 - b^2 - c^2}{-2 \times b \times c})$$

$$B = acos(\frac{b^2 - c^2 - a^2}{-2 \times c \times a})$$

$$C = acos(\frac{c^2 - a^2 - b^2}{-2 \times a \times b})$$

- A、B、C，角度； a、b、c，边长。

问题

- 输入三角形的三个顶点，计算并显示三个角的角度。

运行示例

```
Enter three points: 1 1 6.5 1 6.5 2.5
The three angles are 15.26 90.0 74.74
```

问题

- 输入三角形的三个顶点，计算并显示三个角的角度。

运行示例

```
Enter three points: 1 1 6.5 1 6.5 2.5
The three angles are 15.26 90.0 74.74
```

- 流程

问题

- 输入三角形的三个顶点，计算并显示三个角的角度。

运行示例

```
Enter three points: 1 1 6.5 1 6.5 2.5
The three angles are 15.26 90.0 74.74
```

- 流程
  1. 输入三个顶点的坐标（x 坐标、y 坐标，共 6 个数据）
  4. 输出结果

问题

- 输入三角形的三个顶点，计算并显示三个角的角度。

运行示例

```
Enter three points: 1 1 6.5 1 6.5 2.5
The three angles are 15.26 90.0 74.74
```

- 流程
  1. 输入三个顶点的坐标（x 坐标、y 坐标，共 6 个数据）
  2. 分别计算三条边的长度（a,b,c）
  4. 输出结果

问题

- 输入三角形的三个顶点，计算并显示三个角的角度。

运行示例

```
Enter three points: 1 1 6.5 1 6.5 2.5
The three angles are 15.26 90.0 74.74
```

- 流程
  1. 输入三个顶点的坐标（x 坐标、y 坐标，共 6 个数据）
  2. 分别计算三条边的长度（a,b,c）
  3. 分别计算三个角的角度（A,B,C）
  4. 输出结果

老方法

- #*include* < *cstdlib* >
- #*include* < *ctime* >
- `time(0)` 返回系统当前时间 (作为随机数种子)
- `srand(seed)` 设置随机数种子
- `rand()` 返回一个 `0` 至 `RAND_MAX` 之间的随机数值

- `RAND_MAX` 的值与 `int` 位数有关
- 若随机数种子不变，则其生成的随机数列就不会改变。

产生 [10, 100] 以内的随机数

```
1  #include <iostream>
2  #include<cstdlib>
3  #include<ctime>
4  using namespace std;
5
6  int main( ){
7     srand(time(0));
8     for(int i = 0; i < 10; i++){
9        int x = 10 + rand() % (100 -10 + 1);
10       cout << x <<endl;
11    }
12    return 0;
13 }
```

新方法（≥ C++ 11）

- #*include* < *random* >
- random number engines(引擎类)
  - 负责生成原始随机数
- random number distributions(分布类)
  - 迫使生成的随机数在统计上满足指定的概率分布

新方法

```
1  #include <iostream>
2  #include <random>
3  using namespace std;
4
5  int main( ){
6      default_random_engine e(time(0));//生成随机无符号数
7
8      //生成均匀分布的随机数[10,100]
9      uniform_int_distribution<unsigned> u(10,100);
10
11     for(int i = 0; i < 10; ++i){
12         cout << u(e) << endl;//’u的范围是’’e生成的随机数范围’
13     }
14     return 0;
15 }
```

随机浮点数

```
1    default_random_engine e; //生成无符号的随机整数
2    uniform_real_distribution<double> u(0, 1);//均匀分布[0,1]
3    for(size_t i = 0; i < 10; ++i){
4        cout << u(e) << endl; //从随机整数到随机浮点数的映射
5    }
```

随机浮点数

```
1    default_random_engine e; //生成无符号的随机整数
2    uniform_real_distribution<double> u(0, 1);//均匀分布[0,1]
3    for(size_t i = 0; i < 10; ++i){
4        cout << u(e) << endl; //从随机整数到随机浮点数的映射
5    }
```

- 最常用（但不正确）的做法

随机浮点数

```
1    default_random_engine e; //生成无符号的随机整数
2    uniform_real_distribution<double> u(0, 1);//均匀分布[0,1]
3    for(size_t i = 0; i < 10; ++i){
4        cout << u(e) << endl; //从随机整数到随机浮点数的映射
5    }
```

- 最常用（但不正确）的做法
  - 用 rand() 的结果除以 RADN_MAX，即系统定义的 rand 函数可以生成的最大随机数的上界。
  - 不正确的原因：随机整数的精度通常低于随机浮点数。这导致，有一些浮点数永远不会被产生。

字符

- 字符数据类型表示单个字符
- C++ 采用 ASCII 编码表
- ASCII 表是用于表示大小写英文字母、数字、标点符号和控制字符的 8 位编码表
- char 与 int 互换

```
char letter = 'A';
char numChar = '9';
```

- 单引号包围的单个符号 (或转义字符)

char 与 int 的互换

```
1  #include <iostream>
2  using namespace std;
3  int main( ){
4      char ch = 'c';
5
6      cout << ch << endl;
7      cout << ++ch << endl;
8      cout << ch << endl;
9      cout << ch-- << endl;
10     cout << ch << endl;
11
12     return 0;
13 }
```

字符的比较

- 判断一个字符是否是“大写字母”？

字符的比较

- 判断一个字符是否是“大写字母”?

```
char ch = ' ';
cin >> ch;
if(ch >= 'A' && ch <= 'Z'){
  cout << ch << " 是大写字母" ;
}
```

- 本质上，字符比较是比较字符的 ASCII 码值。

常用字符判断函数 I

```
char ch = ' ';
cin >> ch;

if(isdigit(ch)){
    cout << ch << "是一个数字" << endl;
}
if(isalpha(ch)){
    cout << ch << "是一个字母" << endl;
}
if(isalnum(ch))  cout << ch << "可能是一个字符或者数字" << endl;
```

常用字符判断函数 II

```
if(islower(ch)){
    cout << ch << "是小写字母" << endl;
}
if(isupper(ch)){
    cout << ch << "是大写字母" << endl;
}

char upper_letter = (char)toupper('a');
cout << upper_letter << endl;
```

字符串

- 字符串是由一对双引号包围的符号序列
- string
  - 不是 C++ 基本数据类型
  - 可以把 string 当做一般类型 (int、double、char) 来使用 (+,=,== 等运算符)
- C++ 标准库对 string 的设计思想：让它的行为尽可能像基础类型。
- #*include* < *string* > (在 CodeBlocks 中可省略)

string 变量的简单方法

方法	说明
length()	返回字符串中的字符数
size()	返回字符串中的字符数
at(index)	返回指定位置的字符 (index 从 0 开始)
=	赋值
+=	添加字符串
+	串接两个字符串 (产生一个新的字符串)
关系运算符	比较两个字符串
empty()	判断字符串是否为空
substr()	返回字符串的子串

字符串变量的使用 I

```
1  int main(){
2      string password = "␣";
3      string user_pass = "Tom123456";
4      cin >> password ;
5
6      if(password.empty() == true){
7          cout << "密码为空" << endl;
8      }
9      else if(password.length() < 3){
10         cout << "密码过短" << endl;
11     }
```

字符串变量的使用 II

```
12      else if( password == user_pass ){
13          cout << "密码匹配" << endl;
14      }
15
16      //设置新密码
17      user_pass = password.substr(3,5);
18      cout << user_pass << endl;
19
20      return 0;
21  }
```

读取字符串

- cin 语句
- 空白符（$\backslash t$,$\backslash n$,space）作为数据项间的分隔符

读取字符串

- cin 语句
- 空白符 (\t,\n,space) 作为数据项间的分隔符

```
1  string word;
2  while(cin >> word){
3     cout << ">"
4          << word
5          << endl;
6  }
```

- ctrl + z 结束输入

读取包含空格的整行文本

- `getline` 函数

$$getline(in,\ \ str\_var);$$

读取包含空格的整行文本

- `getline` 函数

$$getline(in,\ \ str\_var);$$

- `in`

读取包含空格的整行文本

- `getline` 函数

$$getline(in,\ \ str\_var);$$

- `in`
  - 数据来源

读取包含空格的整行文本

- `getline` 函数

$$getline(in,\ \ str\_var);$$

- `in`
  - 数据来源
  - `cin`
  - 其他位置 (外部文件)

读取包含空格的整行文本

- `getline` 函数

$$getline(in,\ \ str\_var);$$

- `in`
  - 数据来源
  - `cin`
  - 其他位置 (外部文件)
- `str_var`

读取包含空格的整行文本

- `getline` 函数

$$getline(in,\ \ str\_var);$$

- `in`
  - 数据来源
  - `cin`
  - 其他位置 (外部文件)
- `str_var`
  - 保存读入文本的 `string` 变量

读取包含空格的整行文本

```
string line = " ";

getline(cin, line);

cout << ">" << line << endl;
```

读取以特定符号结尾的整段文本

- `getline` 函数的第二种形式

$$getline(in,\ \ str\_var,\ \ end\_ch);$$

读取以特定符号结尾的整段文本

- getline 函数的第二种形式

$$getline(in,\ \ str\_var,\ \ end\_ch);$$

- end_ch
  - 结束字符
  - 读取的文本中不包含结束字符

读取以特定符号结尾的整段文本

```
string line = " ";

getline(cin, line, '#');

cout << ">" << line << endl;
```

- end_ch　　必须是字符
- 保存在变量 line 中的字符串不含有 # 字符

字符串的常用方法

方法	说明
find()	查找第一个与 value 相等的字符
rfind()	查找最后一个与 value 相等的字符
find_first_of()	查找第一个与 value 中的某值相等的字符
find_last_of()	查找最后一个与 value 中的某值相等的字符
find_first_not_of()	查找第一个与 value 中的任何值都不相等的字符
find_last_not_of()	查找最后一个与 value 中的任何值都不相等的字符

- 返回字符串中“符合查找条件”的第一个字符的索引（数据类型string::size_type）
- 查找不成功（没有找到），返回string::npos。

字符串的查找函数

- 查找函数都采用下面的参数体系：
  - 第一个实参永远是被查找的对象
  - 第二个实参（可有可无）指出 string 内的查找起点（索引）
  - 第三个实参（可有可无）指出欲查找的字符个数（或者说，查找范围、查找长度）
- 返回类型 `string::size_type`
  - string class 定义的一个无符号整数类型
- 查找失败 `string::npos`

字符串和数值的转换

函数	说明
`stoi(str)`	把整数形式的 `str` 转换为一个 `int`
`stod(str)`	把小数形式的 `str` 转换为一个 `double`
`to_string(val)`	把 `val` 转换为一个 `string`

字符串和数值的转换

函数	说明
`stoi(str)`	把整数形式的 `str` 转换为一个 `int`
`stod(str)`	把小数形式的 `str` 转换为一个 `double`
`to_string(val)`	把 `val` 转换为一个 `string`

- `stoi("123")` → `123`
- `stod("3.14")` → `3.14`
- `to_string(123)` → `"123"`

字符串和数值的转换

函数	说明
stoi(str)	把整数形式的 str 转换为一个 int
stod(str)	把小数形式的 str 转换为一个 double
to_string(val)	把 val 转换为一个 string

- stoi(”123”) → 123
- stod(”3.14”) → 3.14
- to_string(123) → ”123”

only $\geq$ C++ 11

字符串和数值互换的另一种方式

```
#include <iostream>
#include <sstream>
using namespace std;
int main(){
    stringstream ss;
    ss << "123 3.14";

    int x = 0;
    double y = 0.0;

    ss >> x >> y;

    cout << "x = " << x << endl << "y = " << y << endl;
    return 0;
}
```

把文件名的扩展名改为 tmp I

```
1  #include<iostream>
2  #include<string>
3  using namespace std;
4
5  int main(){
6      string filename, basename, extname, tmpname;
7      const string suffix("tmp");
8
9      cin >> filename; //输入文件名
10     string::size_type idx = filename.find('.');
11     if(idx == string::npos){//没有找到点号即没有扩展名。,
12         tmpname = filename + '.' + suffix;
```

把文件名的扩展名改为 tmp II

```
13      }
14      else{//存在点号
15          basename = filename.substr(0, idx);
16          extname = filename.substr(idx + 1);
17          if(extname.empty() == true){//扩展名为空，只有点号。
18              tmpname = filename;
19              tmpname += suffix;
20          }
21          else if(extname == suffix){//原扩展名就是tmp
22              tmpname = filename;
23              tmpname.replace(idx + 1, extname.size(), "xxx");
24          }
```

把文件名的扩展名改为 *tmp* III

```
25          else{//常规处理用,替换原扩展名。tmp
26              tmpname = filename;
27              tmpname.replace(idx + 1, string::npos, suffix);
28          }
29      }
30
31      cout << filename << "␣=>␣" << tmpname << endl;
32
33      return 0;
34  }
```

Q & A