搜索话题、问题或人... 提问 首页 话题 发现 消息 ywang

程序员 299 问题 编程 计算机科学 修改

有没有一段代码，让你觉得人类的智慧也可以璀璨无比？ 修改

不一定要是完整算法，就是那种看着看着就觉得嗨爆了，惊为天人的结构或语句。 修改

1 条评论 分享 • 邀请回答 举报

142 个回答

按投票排序

烧茄子，梦想是成为百科全书一样的人（。•`…

llUnbiri、e KING、陆生 等人赞同

当然是这个啦！

用三段 140 字符以内的代码生成一张 1024×1024 的图片

原文 by Matrix67

> Kyle McCormick 在 StackExchange 上发起了一个叫做 Tweetable Mathematical Art 的比赛，参赛者需要用三条推这么长的代码来生成一张图片。具体地说，参赛者需要用 C++ 语言编写 RD 、 GR 、 BL 三个函数，每个函数都不能超过 140 个字符。每个函数都会接到 i 和 j 两个整型参数（0 ≤ i, j ≤ 1023），然后需要返回一个 0 到 255 之间的整数，表示位于 (i, j) 的像素点的颜色值。举个例子，如果 RD(0, 0) 和 GR(0, 0) 返回的都是 0 ，但 BL(0, 0) 返回的是 255 ，那么图像的最左上角那个像素就是蓝色。参赛者编写的代码会被插进下面这段程序当中（我做了一些细微的改动），最终会生成一个大小为 1024×1024 的图片。

```
// NOTE: compile with g++ filename.cpp -std=c++11

#include <iostream>
#include <cmath>
#include <cstdlib>
#define DIM 1024
#define DM1 (DIM-1)
#define _sq(x) ((x)*(x)) // square
#define _cb(x) abs((x)*(x)*(x)) // absolute value of cube
#define _cr(x) (unsigned char)(pow((x),1.0/3.0)) // cube root

unsigned char GR(int,int);
unsigned char BL(int,int);

unsigned char RD(int i,int j){
// YOUR CODE HERE
}
unsigned char GR(int i,int j){
// YOUR CODE HERE
}
unsigned char BL(int i,int j){
// YOUR CODE HERE
}

void pixel_write(int,int);
FILE *fp;
int main(){
fp = fopen("MathPic.ppm","wb");
fprintf(fp, "P6\n%d %d\n255\n", DIM, DIM);
for(int j=0;j<DIM;j++)
for(int i=0;i<DIM;i++)
pixel_write(i,j);
fclose(fp);
return 0;
}
void pixel_write(int i, int j){
static unsigned char color[3];
color[0] = RD(i,j)&255;
color[1] = GR(i,j)&255;
color[2] = BL(i,j)&255;
fwrite(color, 1, 3, fp);
}
```

> 我选了一些自己比较喜欢的作品，放在下面和大家分享。

首先是一个来自 Martin Büttner 的作品：

它的代码如下：

```
unsigned char RD(int i,int j){
return (char)(_sq(cos(atan2(j-512,i-512)/2))*255);
}

unsigned char GR(int i,int j){
return (char)(_sq(cos(atan2(j-512,i-512)/2-2*acos(-1)/3))*255);
}

unsigned char BL(int i,int j){
return (char)(_sq(cos(atan2(j-512,i-512)/2+2*acos(-1)/3))*255);
}
```

同样是来自 Martin Büttner 的作品：

这是目前暂时排名第一的作品。它的代码如下：

```
unsigned char RD(int i,int j){
#define r(n)(rand()%n)
static char c[1024][1024];return!c[i][j]?c[i][j]=!r(999)?r(256):RD((i+r(2))%1024,(j+r(2))%1024):c[i][j];
}

unsigned char GR(int i,int j){
static char c[1024][1024];return!c[i][j]?c[i][j]=!r(999)?r(256):GR((i+r(2))%1024,(j+r(2))%1024):c[i][j];
}

unsigned char BL(int i,int j){
static char c[1024][1024];return!c[i][j]?c[i][j]=!r(999)?r(256):BL((i+r(2))%1024,(j+r(2))%1024):c[i][j];
}
```

下面这张图片仍然出自 Martin Büttner 之手：

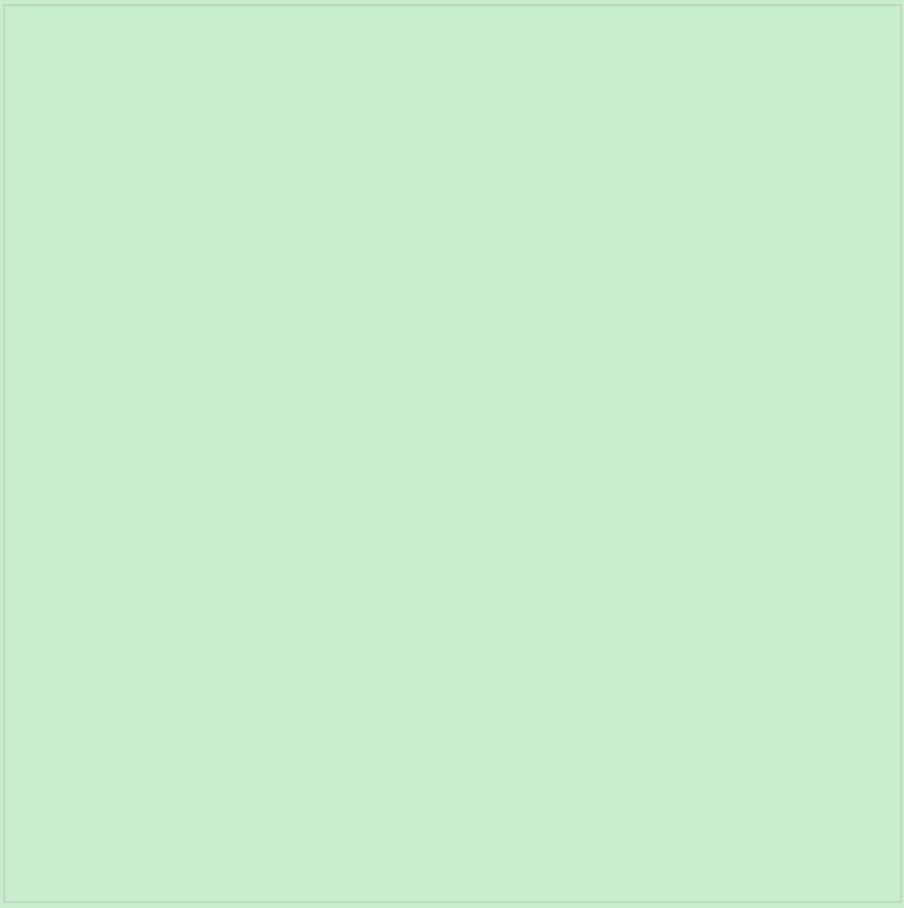
难以想象， Mandelbrot 分形图形居然可以只用这么一点代码画出：

```
unsigned char RD(int i,int j){
float x=0,y=0;int k;for(k=0;k++<256;){float a=x*x-y*y+(i-768.0)/512;y=2*x*y+(j-512.0)/512;x=a;if(x*x+y*y>4)break;}return log(k)*47;
}

unsigned char GR(int i,int j){
float x=0,y=0;int k;for(k=0;k++<256;){float a=x*x-y*y+(i-768.0)/512;y=2*x*y+(j-512.0)/512;x=a;if(x*x+y*y>4)break;}return log(k)*47;
}

unsigned char BL(int i,int j){
float x=0,y=0;int k;for(k=0;k++<256;){float a=x*x-y*y+(i-768.0)/512;y=2*x*y+(j-512.0)/512;x=a;if(x*x+y*y>4)break;}return 128-log(k)*23;
}
```

> Manuel Kasten 也制作了一个 Mandelbrot 集的图片，与刚才不同的是，该图描绘的是 Mandelbrot 集在某处局部放大后的结果：



> 它的代码如下：

```
unsigned char RD(int i,int j){
double a=0,b=0,c,d,n=0;
while((c=a*a)+(d=b*b)<4&&n++<880)
{b=2*a*b+j*8e-9-.645411;a=c-d+i*8e-9+.356888;}
return 255*pow((n-80)/800,3.);
}

unsigned char GR(int i,int j){
double a=0,b=0,c,d,n=0;
while((c=a*a)+(d=b*b)<4&&n++<880)
{b=2*a*b+j*8e-9-.645411;a=c-d+i*8e-9+.356888;}
return 255*pow((n-80)/800,.7);
}

unsigned char BL(int i,int j){
double a=0,b=0,c,d,n=0;
while((c=a*a)+(d=b*b)<4&&n++<880)
{b=2*a*b+j*8e-9-.645411;a=c-d+i*8e-9+.356888;}
return 255*pow((n-80)/800,.5);
}
```

> 这是 Manuel Kasten 的另一作品：

生成这张图片的代码很有意思：函数依靠 static 变量来控制绘画的进程，完全没有用到 i 和 j 这两个参数！

```
unsigned char RD(int i,int j){
static double k;k+=rand()/1./RAND_MAX;int l=k;l%=512;return l>255?511-l:l;
}

unsigned char GR(int i,int j){
static double k;k+=rand()/1./RAND_MAX;int l=k;l%=512;return l>255?511-l:l;
}

unsigned char BL(int i,int j){
static double k;k+=rand()/1./RAND_MAX;int l=k;l%=512;return l>255?511-l:l;
}
```

这是来自 githubphagocyte 的作品：

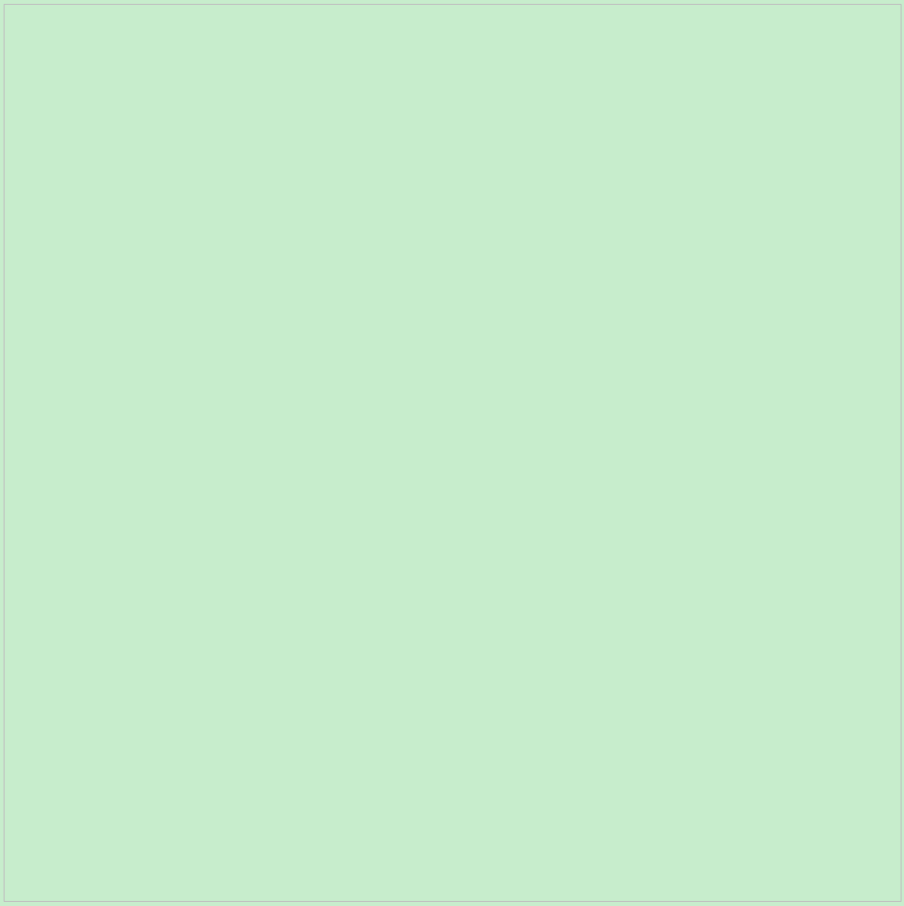
它的代码如下：

```
unsigned char RD(int i,int j){
float s=3./(j+99);
float y=(j+sin((i*i+_sq(j-700)*5)/100./DIM)*35)*s;
return (int((i+DIM)*s+y)%2+int((DIM*2-i)*s+y)%2)*127;
}
```

```
unsigned char GR(int i,int j){
float s=3./(j+99);
float y=(j+sin((i*i+_sq(j-700)*5)/100./DIM)*35)*s;
return (int(5*((i+DIM)*s+y))%2+int(5*((DIM*2-i)*s+y))%2)*127;
}

unsigned char BL(int i,int j){
float s=3./(j+99);
float y=(j+sin((i*i+_sq(j-700)*5)/100./DIM)*35)*s;
return (int(29*((i+DIM)*s+y))%2+int(29*((DIM*2-i)*s+y))%2)*127;
}
```

> 这是来自 githubphagocyte 的另一个作品：



> 这是一张使用 diffusion-limited aggregation 模型得到的图片，程序运行起来要耗费不少时间。代码很有意思：巧妙地利用宏定义，打破了函数与函数之间的界限，三段代码的字数限制便能合在一起使用了。

```
unsigned char RD(int i,int j){
#define D DIM
#define M m[(x+D+(d==0)-(d==2))%D][(y+D+(d==1)-(d==3))%D]
#define R rand()%D
#define B m[x][y]
return(i+j)?256-(BL(i,j))/2:0;
}

unsigned char GR(int i,int j){
#define A static int m[D][D],e,x,y,d,c[4],f,n;if(i+j<1){for(d=D*D;d;d--){m[d%D][d/D]=d%6?0:rand()%2000?1:255;}for(n=1
return RD(i,j);
}

unsigned char BL(int i,int j){
A;n;n++){x=R;y=R;if(B==1){f=1;for(d=0;d<4;d++){c[d]=M;f=f<c[d]?c[d]:f;}if(f>2){B=f-1;}else{++e%=4;d=e;if(!c[e]){B=0;M=1;}}}}}return m[i][j];
}
```

> 最后这张图来自 Eric Tressler：

这是由 logistic 映射得到的 Feigenbaum 分岔图。和刚才一样，对应的代码也巧妙地利用了宏定义来节省字符：

```
unsigned char RD(int i,int j){
#define A float a=0,b,k,r,x
#define B int e,o
#define C(x) x>255?255:x
#define R return
#define D DIM
R BL(i,j)*(D-i)/D;
}

unsigned char GR(int i,int j){
#define E DM1
#define F static float
#define G for(
#define H r=a*1.6/D+2.4;x=1.0001*b/D
R BL(i,j)*(D-j/2)/D;
}

unsigned char BL(int i,int j){
F c[D][D];if(i+j<1){A;B;G;a<D;a+=0.1){G b=0;b<D;b++){H;G k=0;k<D;k++){x=r*x*(1-x);if(k>D/2){e=a;o=(E*x);c[e][o]+=0.01;}}}}}R C(c[j][i])*i/D;
}
```

编辑于 2015-05-23　　139 条评论 • 作者保留权利

高城，每一天都很重要

决明、杨马宇、文盲君 等人赞同

我在这里提供我见识到的三个精彩算法的解析，强烈地推荐给初学的算法爱好者，它们可能会令你眼界大开，同时坚定你在算法大道上勇往直前的信念。

#3. 二进制是人类的好朋友，在线的树的最近公共祖先（LCA）算法：

利用数的二进制表示可以产生很多加速算法，online-LCA是其中之一。许多算法的加速是对数率的，就是利用了数的二进制表示。

首先定义二维数组：prede[N+1][B+1]，N表示树的结点的数量，结点以数字1到N代指，B满足条件：2^(B)>=N

令fa[i]表示结点i的父结点，那么prede[i][b]的含义是：

```
prede[i][0] = fa[i];
prede[i][b] = prede[prede[i][b-1]][b-1]; // b >= 1
```

也就是说，prede[i][b]指的是从结点i往上走2^(b)步，所到达的结点。如果走到了尽头，就令prede[i][b]为0。

我们只需要O(NlogN)的复杂度，就可以完成prede的初始化。此外，我们还需要预处理出所有结点的高度，也就是depth[i]，定义为：

```
depth[root] = 0;
depth[i] = depth[fa[i]] + 1;
```

当遇到询问LCA(x, y)，我们只需要采取如下行动，就可以O(logN)的代价获得答案：

```
int lca(int x, int y) {
    if (depth[x] > depth[y]) swap(x, y);
    for(int i = B; i >= 0; i --){
        //令x和y高度一致
        if (depth[prede[y][i]] >= depth[x]) y = prede[y][i];
    }
    //注意此时有可能出现x == y，那么LCA(x,y) == x，下方的for
    //就不起作用了。
    for(int i = B; i >= 0; i --){
       //如果prede[x][i]和prede[y][i]不相同，说明这两者的高度
       //都大于所求的LCA(x,y)，也就是在LCA(x,y)的下方，此时令
       //x和y一同往根部以2^(i)的步数爬升
       if (prede[x][i] != prede[y][i]) x = prede[x][i], y = prede[y][i];
    }
    if (x == y) return x;   //此时LCA（x,y）= x
    return prede[x][0];     //此时x和y有共同的父结点
}
```

上述代码的精髓在于两个for(int i = B; i >= 0; i --)，这里利用了数的二进制表示。可以证明，对于任何严格小于2^(B+1)的非负整数t，下面的代码运行之后可以令a == t，

```
int a = 0;
for(int i = B; i >= 0; i --){
    if(a + (1<<i) <= t) a += (1<<i);
}
```

#2. 集合之交，树状数组，动态更改、查询数组前缀和算法。

实现树状数组所需的代码极为简易，实际上它是一棵残缺的线段树，它可以实现一部分线段树的功能（但凡可以化为区间求和的问题基本上都能解决），但是毕竟不如线段树功能完整，有兴趣的读者应该学习一下线段树的知识。

问题描述：利用预处理的前缀和数组pre[N + 1]，我们可以O(1)的代价对静态的数组A[N + 1]求取区间和：

```
pre[i] = A[0] + A[1] + A[2] + ... + A[i];
A[a] + A[a+1] + A[a+2] + ... + A[b] = pre[b] - pre[a-1];
```

但是当需要对数组A进行动态的更改时，上述代码就失效了。我们需要一种算法，可以动态地更改以及查询前缀和数组pre[N+1]。下面首先展示树状数组的代码，然后解释其数学原理，它的插入和查询的代价都是O(logN)：

```
int Count[BiggestN+1], N; //使用前令Count所有元素为0，规定A[0]没有数
                          //据，也就是说数据从A[1]开始存，pre[0]总为零

//实现功能A[i] += add
void insert(int i, int add)
{
    while( i <= N )
    {
        Count[i] += add;
        i += i&(-i);
    }
}
//返回pre[i]的值
int query(int i)
{
    int num = 0;
    while( i > 0 )
    {
        num += Count[i];
        i -= i&(-i);
    }
```

```
    return num;
}
```

算法中最关键的语句是位操作i&(-i)，读者在稿纸上算一算就可以知道：
i -= i&(-i)的功能是令i的最低的非0位变为0；
i += i&(-i)的功能是令i的最低的非0位变为0，并往更高一位进一。

理解树状数组的行为，需要构造两个集合：

Define lowbit(i) = i&(-i);
up(a) = {a, a1, a2, ...}, ai = a(i-1) + lowbit(a(i-1));
down(a) = {a, a1, a2, ..., 0}, ai = a(i-1) - lowbit(a(i-1));

可以证明，对于任何a <= b的正整数对(a,b)，up(a)和down(b)的交集都有且仅有一个元素。对这个定理进行含糊的说明是很容易的，a == b的情况不必考虑，a < b时，总有一个最大的i，使得b的第i位大于a的第i位（也就是b的第i位是1，而a的第i位是0），那么对b产生down(b)，对a产生up(a)，它们的唯一交集就是(1<<i)。注意这里讨论的第i位的i是从0开始索引的。读者可以在稿纸上找若干数对进行实验来加深印象。

有了上述定理，我们就不难意会insert函数和query函数的作用了。

#1. 机器浮点数的秘密，"巧夺天工"的完美实例，基于标准浮点数的快速开平方倒数算法

这是一个公开的秘密，这是一个所有程序员得以欣赏的智慧之美。她在许多程序员的心目中高居“最美代码”的第一位，所有溢美之词都无法表达他们所感受到的震撼。

一定会有许多人想在这里贴这段代码，少年，来我这里，我帮你揭开她神秘的面纱。

公式太多，贴图讲解。

Vote Me Up If You Like My Answer ^_^

编辑于 2015-05-21　25 条评论 • 作者保留权利

Voyga， voyga.top

Zover、龚健威、不吃烧饼饿得慌 等人赞同

```
ﾟωﾟ/= / `m´) / ~┻━┻   //*´▽`*/ ['_']; o=(ﾟｰﾟ)  =_=3; c=(ﾟΘﾟ) =(ﾟｰﾟ)-(ﾟｰﾟ); (ﾟДﾟ) =(ﾟΘﾟ)= (o^_^o)/ (o^_^o);(ﾟДﾟ)={ﾟΘﾟ: '_' ,ﾟωﾟ/ : ((ﾟ
```

这是一段js，大家可以试一下。

发布于 2015-07-29　40 条评论 • 作者保留权利

Argentum， OIer/会一点吉他/高复狗/不爱读书/节操掉…

雷力明、闲圣、林悦 等人赞同

```
int find(int x){
       return x==Fa[x]?x:Fa[x]=find(Fa[x]);
}
```

一行并查集

不管从算法内涵到代码表示都简直精致优雅到炸

编辑于 2015-05-13　25 条评论 • 作者保留权利

毛草，软件工程师/MBA在读/动漫宅/单身/万能

Max Xing、JackAlan、Kou zen 等人赞同

```
#define _ -F<00||--F-00--;
int F=00,00=00;main(){F_00();printf("%1.3f\n",4.*-F/00/00);}F_00()
{
           _-_-_-_
       _-_-_-_-_-_-_-_
    _-_-_-_-_-_-_-_-_-_-_
  _-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_
  _-_-_-_-_-_-_-_-_-_-_-_-_
    _-_-_-_-_-_-_-_-_-_-_
       _-_-_-_-_-_-_-_
           _-_-_-_
}
```

用一段形状是圆形的程序，通过计算代码本身的面积，计算圆周率的近似值

来源：International Obfuscated C Code Contest
ioccc.org/1988/westley....

解释：
每一行的第一个_会作为

```
-F < 0 || --F<00--;
```

执行。由于执行过程中F永远小于0，所以-F<0对每一行不成立。于是||左边不成立，右边的表达式--F<00--便会被执行。由于结果并没有意义，实际上被执行的只有--F和00--两条语句。

每一行的其他部分都是若干个-_。根据作者的解释，这部分会被作为

```
--F < 0 || --F < 00--;
```

执行。由于F一直小于0，--F<0在本程序中永远成立。所以右半部份不会被执行。也就是这一行指令只相当于执行--F。

因此，在本程序执行完毕后，--F的执行次数为_出现的次数，也就是该圆的面积（因为该程序的代码布局），00--的执行次数为该圆的直径（因为每行执行一次）。

最后的printf中计算的表达式是4.*F/00/00，也就是4*面积/直径/直径=4*面积/(4*半径^2)=面积/半径^2，也就是圆周率的值。

编辑于 2015-05-13　25 条评论 • 作者保留权利

栈堆，反击人生对人生观开的玩笑

闲圣、阮永顺、东东 等人赞同

大一上编程导论，最后一个project老师让我们用Java写一个能print itself的程序。。查阅wiki最后千辛万苦写出来以后觉得真是。。智慧。。
参见 Quine (computing)

发布于 2015-05-13　19 条评论 • 作者保留权利

何亮，roba

闲圣、钟建华、田乔 等人赞同

```
int find(p) {
  for (int i : adjacent[p]) {
    if (!cover[i]) {
      int q = link[i];
      link[i] = p;
```

```
      cover[i] = true;
      if (q == -1 || find(q)) return 1;
      link[i] = q;
    }
  }
  return 0;
}

int bimatch() {
  int total = 0;
  memset(link, -1, sizeof(link));
  for (int i = 0 ; i < nLeft; ++i) {
    memset(cover, false, sizeof(cover));
    total += find(i);
  }
  return total;
}
```

上面代码实现了求二分图最大匹配的匈牙利算法。当时看懂这段代码后我惊呆了。完全没有炫技，就是简单直白地把该写的写出来，美得如此自然。

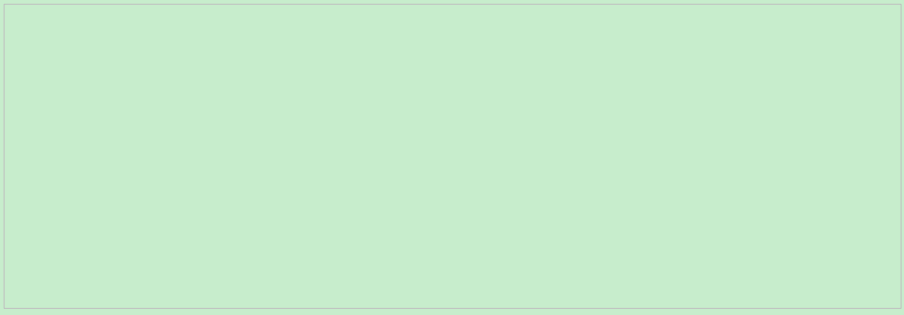
发布于 2015-07-21　43 条评论 • 作者保留权利

没什么大不了

许俊彬、周浩、山风 等人赞同

真心佩服PID！！！暂时绝对是我觉得最惊艳的东西了。
经典的东西往往是简单的！而PID控制算法恰好就符合了这一点，短短一个公式，十多航。
在工业应用中，PID及其衍生算法几乎是应用最广泛的算法。而它问世的六十、七十年以来，一直在使用，仍未淘汰。至于在我们学生当中，假如对PID**足够熟悉**，对PID的参数整定**熟练**的话，我觉得参加电子设计（控制类），飞思卡尔智能车，机器人比赛等这类的比赛，获奖的压力肯定会小很多很多。其他类型的创新项目只要涉及到控制类的，问题也应该不大了吧。至于其他领域，我就不太清楚了，毕竟也没太多接触。

下面来大概介绍下这个算法。
首先，这些介绍都是之前做比赛或是电子设计时，从网上搜集到的各种资料（仔细看完应该可以大概入门了吧）。 所以假如有侵犯个人版权之类的，请联系我。我会立刻改正，并补上作者。毕竟是第一次完整答题，有什么问题，请大家提出，谢谢~
不废话，进入正题，开启搬运工模式
****
第一，就是这张图了



PID 对应的就是 比例（Proportion）、积分（Integration）、微分（Differentiation），对应着上面的图，意思就是。通过一个输入量（偏差值），经过比例、积分、微分、的运算后（应用时不一定需要三个步骤都有）得到一个输出量（结果）。再把这个输出量直接给执行部件，执行操作。此后通过，传感器得到实际值时，再于目标值作差，传给输入量。接着，一直循环下去。直到你所需要控制的那个量慢慢逼近，达到你设定的目标值为止。
最简单的例子就是，利用PWM，编码器的反馈量来对小车的车速形成一个闭环控制。

连续形式的时域表达式

而我们实际实用时，只需要在公式中定义 、 、 这三个变量。
(实在实在不好意思，后天有一门考试。现在还没好好复习完.....考完再好好整理更新，行嘛？)
下面是一段不完整的程序，还未整理。中间的motorgun,对应的就是PID这个公式。直接可以把一个角度传感器传进来的角度值 通过计算得到一个数字后，把它赋给电机的PWM值。随后经过多次运算，实际角度值就能慢慢接近目标角度值。

```
void PID (void)
{
 NOWPOS = GET_POS();    // 通过传感器获得当前值
 MOTOGUN = 0;           //
 pid.LastError = NOWPOS - pid.SetPoint; // 用当前值减去目标的，得到输入的那个偏差
        pid.SumError+= pid.LastError;   // 中间积分项的累加
```

```
        MOTOGUN = pid.P*pid.LastError +
                  pid.I*pid.SumError +
                  pid.D* (pid.LastError - pid.PrevError); //此处是PID的公式！！
        pid.PrevError = pid.LastError;
 MOTOR(MOTOGUN);                  //直接把计算值给了电机PWM程序
}
```

我先提到这，假如大家对PID特别感兴趣，可以先到网上查找。实在不好意思~~

编辑于 2015-05-13　37 条评论 • 作者保留权利

刘巍然-学酥， PhD/PKE/Java/听译

FrozenYogurtPuff、帽子沐、陈业明 等人赞同

0 引言

这个题我已经留坑很久很久了，但是最近忙于赶一篇论文，5月31日是投稿截止日期。今天终于是把论文写完了，于是准备把留的坑都填一填。

前辈们已经把很多经典的、让人眼前一亮的代码展示出来了。我虽然不是码农，但也经常会看一些奇怪的代码。这里我列举五个我觉得最有趣、最好玩的代码给大家。这些代码一部分已经在之间的回答给出了，但为了Self-Contained，我也把这些已经给出的代码再次展示出来，并给出原始答案的链接、作者以及出处。

前方多图预警，超长答案预警，建议在WiFi环境下观看。

==========

1. 真正改变世界的算法：向杰出的数学和计算机科学家们致敬

真正改变世界的是那些依托于数学的，真正高大上的代码。知乎er @张狗狗提到的“支配世界的10个算法”（The real 10 algorithms that dominate our world，The real 10 algorithms that dominate our world ）就是10个这类典型的算法。让我惊喜的是，这10个算法中竟然有4个与密码学相关（RSA算法，安全Hash算法，整数分解算法，随机数生成算法）。难道说真正掌控世界的是密码学嘛（笑）。作为一个密码学研究者，我感觉很激动啊！具体的内容 知乎er@张狗狗已经回答的非常详细了，请大家参考他的答案（有没有一段代码，让你觉得人类的智慧也可以璀璨无比？ - 张狗狗的回答）。作为致敬，我仅列出这10个算法，并请大家一起像这些杰出的科学家们致敬。

- 三种排序算法。**归并排序**（Mergesort），提出者：约翰·冯·诺尼曼（John von Neumann），现代计算机创始人；**快速排序**（Quicksort），提出者：托尼·霍尔（Tony Hoare），1980年图灵奖获得者；**堆排序**（Heapsort），基于计算机基本数据结构堆（Heap）的排序方法。
- **傅里叶变换**（Fourier Transform），提出者：约瑟夫·傅立叶（Joseph Fourier），温室效应的发现者，名字被刻在埃菲尔铁塔的72位法国科学家与工程师的其中一位。小行星10101号以傅里叶命名；**快速傅里叶变换**（Fast Fourier Transform），原始提出者：约翰·卡尔·弗里德里希·高斯（Carolus Fridericus Gauss），历史上最重要的数学家之一，被誉为“数学王子”。
- **Dijkstra最短路径算法**（Dijkstra Shortest Path Algorithm）。提出者：艾兹赫尔·戴克斯特拉（Edsger Wybe Dijkstra），1972年图灵奖获得者。
- **RSA公钥密码体制**（RSA Public Key Cryptosystem）。提出者：罗纳德·李维斯特（Ron Rivest）、阿迪·萨莫尔（Adi Shamir）和伦纳德·阿德曼（Leonard Adleman）。他们为2002年图灵奖获得者。
- **安全Hash算法**（Secure Hash Algorithm）。一些经典的安全Hash算法及其提出者们。MD5算法，提出者：罗纳德·李维斯特（Ron Rivest），2002年图灵奖获得者。SHA家族（SHA-1，SHA-224，SHA-256，SHA-384，SHA-512），提出者：美国国家安全局（National Security Agency）。值得注意的是，MD5以及SHA-1算法在2005年由中国密码学家王小云、殷益群、于红波从理论上破解。王小云因“国际通用Hash函数的破解”获颁陈嘉庚科学奖信息技术科学奖。
- **整数分解**（Integer Factorization）。该问题已可以在量子计算机上以多项式复杂度解决，被称为秀尔算法（Shor Algorithm），提出者：彼得·威廉·秀尔（Peter Williston Shor），1998年奈望林纳奖（Nevanlinna Prize，计算机科学的数学方面重要贡献奖，每4年一届，得奖者在获奖当年不得大于40岁）获得者。
- **链接分析**（Link Analysis）。提出者：Gabriel Pinski、Francis Narin。
- **比例积分微分算法**（Proportional Integral Derivative Algorithm）。提出者：尼古拉斯·米诺尔斯基（Nicolas Minorsky）。
- **数据压缩算法**（Data Compression Algorithms）。包括数据压缩、图像压缩等等。
- **随机数生成器**（Random Number Generation）。

==========

2. 数学拯救计算机科学算法：Quake III源代码中的玄机

很多知乎er们都提到了一个神奇的数0x5f3759df（ @高城 ， @陆天培 等）。追溯这个数的来源，我们就要谈到一个经典游戏Quick III了。Quick III从现在的眼光看来实在不是一个很精细的游戏。但是在当时，Quick III的画质简直是狂暴酷炫掉渣天。在Quick系列，Doom系列，及其3D引擎出现之前，有谁能想到计算机也可以显示如此神奇的3D效果呢？

这个3D引擎的开发者是约翰·卡马克（John Carmack）。他是美国电子游戏程序员，ID Software（一个著名游戏开发公司）的创始人之一。他在1999年，被时代杂志评选为科技领域50大影响力人物榜单，并列名列第10位。为表彰他的在电子游戏和电视游戏领域所作出的杰出贡献，卡马克称为第四位进入互动艺术和科学学院（The Academy of Interactive Arts and Science）名人堂的人物。我们来看看其余人物的名字吧（美国互动艺术与科学学院（AIAS）名人堂 smile 新浪博客 ）：

- 宫本茂：任天堂公司，《超级马里奥兄弟》的创始人。
- 丹妮·邦顿·贝瑞：名人堂游戏《M·U·L·E》的制作人。
- 席德·梅尔：FiraxisGames公司，《文明》系列的制作人。
- 铃木裕：世嘉公司，众多街机游戏的制作人。
- 威尔·赖特：Maxis公司，《模拟城市》系列的制作人。
- 坂口博信：Square Enix公司，《最终幻想》系列的制作人。
- 理查德·加略特：NCsoft / DestinationGames公司，《创世纪》系列的制作人。
- 彼得·莫利纽克斯：牛蛙公司，《上帝也疯狂》，《主题公园》，《主题医院》，《地牢守护者》制作人。
- 特里普·霍金斯：3DO公司，《魔法门》系列，《英雄无敌》系列，《玩具兵》系列制作人。
- 麦克：暴雪公司，《魔兽世界》制作人。

卡马克在计算机领域最重要的贡献是，极大限度地压榨计算机计算性能，实现了一系列令人惊叹的数学函数。其中最知名的就是很多知乎er们提到的开平方取倒数算法，这个算法的基本原理是牛顿迭代法。原文出处来自一个Sqrt函数引发的血案 。

```
float Q_rsqrt( float number )
{
    long i;
    float x2, y;
    const float threehalfs = 1.5F;

    x2 = number * 0.5F;
    y  = number;
    i  = * ( long * ) &y;
    i  = 0x5f3759df - ( i >> 1 ); // what the fuck?
    y  = * ( float * ) &i;
    y  = y * ( threehalfs - ( x2 * y * y ) ); // 1st iteration
    // y  = y * ( threehalfs - ( x2 * y * y ) ); // 2nd iteration, this can be removed

    #ifndef Q3_VM
    #ifdef __linux__
        assert( !isnan(y) ); // bk010122 - FPE?
    #endif
    #endif
    return y;
}
```

看看这段代码吧，只用了2次叠代，而且其实根本上就没用叠代，因为算一次直接就得到结果了。这个函数的效率比C库函数中执行sqrt()函数再取倒数要快4倍。

为什么能得到如此令人惊叹的效果呢？这段代码最让人费解的就是下面这行代码：

```
i  = 0x5f3759df - ( i >> 1 ); // what the fuck?
```

这个0x5f3759df是个什么鬼？牛顿迭代法的原理是先猜测一个值，然后从这个值开始进行叠代。因此，猜测的值越准，叠代的次数越少。卡马克选了0x5f3759df这个值作为猜测的结果，再加上后面的移位算法，得到的y非常接近1/sqrt(n)。这样，我们只需要2次牛顿迭代法就可以达到我们所需要的精度。

> 普渡大学的数学家Chris Lomont看了以后觉得有趣，决定要研究一下卡马克弄出来的这个猜测值有什么奥秘。在精心研究之后，Lomont从理论上也推导出一个最佳猜测值，和卡马克的数字非常接近，0x5f37642f。Lomont计算出结果以后非常满意，于是拿自己计算出的起始值和卡马克的神秘数字做比赛，看看谁的数字能够更快更精确的求得平方根。结果是卡马克赢了...
>
> Lomont怒了，采用暴力方法一个数字一个数字试过来，终于找到一个比卡马克数字要好上那么一丁点的数字，虽然实际上这两个数字所产生的结果非常近似，这个暴力得出的数字是0x5f375a86。
> Lomont为此写下一篇论文，"Fast Inverse Square Root"（这里原文给出的地址有误，论文的地址为 files.sauliaus.info/Inv ... ）。

感谢开源GPL协议，Quick III的源代码已经公开，我们可以目睹一下Quick III的风采了。源代码下载地址为：ftp://ftp.idsoftware.com/idstuff/source/quake3-1.32b-source.zip 。这个地址需要翻墙才能下载。而那个传说中的代码，位于game/code/q_math.c中。

==========

3. 二进制的神奇应用：基于位运算的代码系列

看到了Quick III中神奇的代码，想必知乎er们也对二进制和位运算的神奇有了一些感觉了。我最开始觉得二进制运算很有趣，是因为Leetcode中的一道题：Single Number（Single Number ）。题目如下：

> Given an array of integers, every element appears *twice* except for one. Find that single one.

问题在于，如何在线性复杂度下实现呢？这里就要用到与或运算了。

二进制运算还有哪些神奇的应用呢？Stanford的一个Ph.D学生Sean Eron Anderson（Sean Anderson ）自己维护了一个网站，里面列举了一大堆二进制运算的神奇应用，链接为：Bit Twiddling Hacks 。

我们来简单看几个运算：

1. 计算一个整数的符号：

```
int v;      // we want to find the sign of v
int sign;   // the result goes here

// CHAR_BIT is the number of bits per byte (normally 8).
sign = -(v < 0);  // if v < 0 then -1, else 0.
// or, to avoid branching on CPUs with flag registers (IA32):
sign = -(int)((unsigned int)((int)v) >> (sizeof(int) * CHAR_BIT - 1));
// or, for one less instruction (but not portable):
sign = v >> (sizeof(int) * CHAR_BIT - 1);
```

2. 检测两个整数的符号是否一致：

```
int x, y;               // input values to compare signs
bool f = ((x ^ y) < 0); // true iff x and y have opposite signs
```

3. 计算一个整数的绝对值：

```
int v;           // we want to find the absolute value of v
unsigned int r;  // the result goes here
int const mask = v >> sizeof(int) * CHAR_BIT - 1;

r = (v + mask) ^ mask;
```

@高城还提到了Google Code上面的一个基于浮点数标准的快速算法库，里面也用到了各种二进制运算，链接为：fastapprox - Fast approximate functions 。

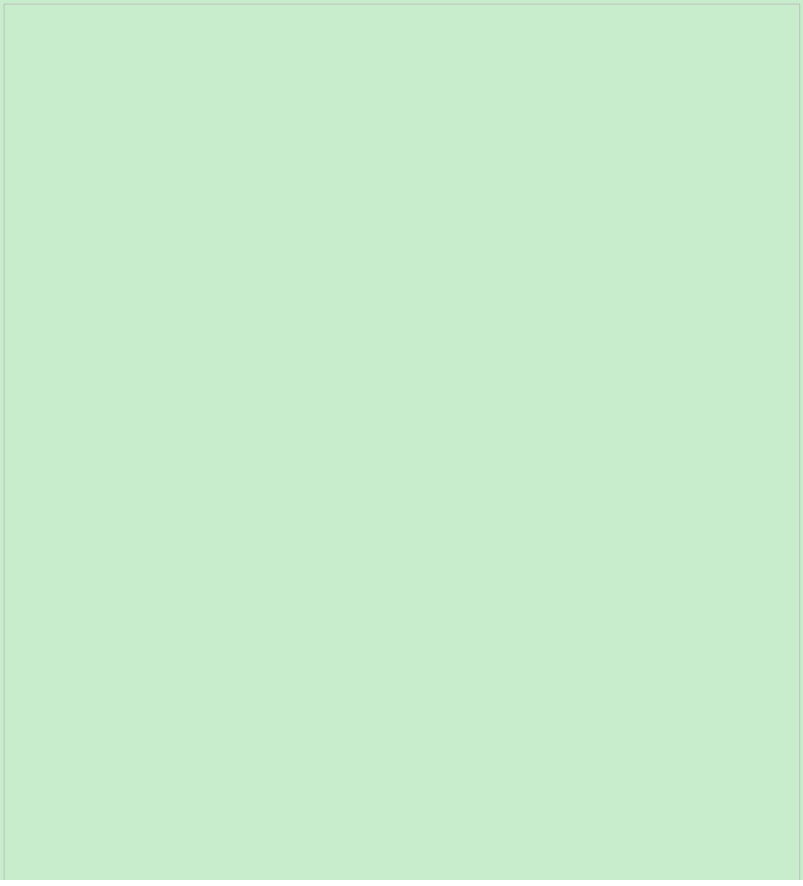
==========

4. 能不能把代码写的够混乱：世界混乱C代码大赛

后面的几个就是比较好玩的东西了。首先我们来看看计算机学科的怪胎们是怎么玩坏C语言的。 @毛草提到了一个神奇的代码，通过计算代码本身的面积，来计算圆周率的近似值。这个代码来自世界混乱C代码大赛（International Obfuscated C Code Contest，IOCCC）。这个大赛的宗旨是，在C语言本身可以执行的条件下，看谁能写出最有创意的最让人难以理解的C语言代码。代码的长度要求限制在4kb以内。大赛的官方链接为：The International Obfuscated C Code Contest 。

第23届IOCCC竞赛已经在2014年10月27日结束，但是我似乎找不到源代码。我们来看看第22届IOCCC竞赛中的代码中又没有什么好玩的东西吧。因为有些代码知乎上显示的不太好，我用截图的形式给出。

Best Painting Tool（ ioccc.org/2013/birken/h ... ）

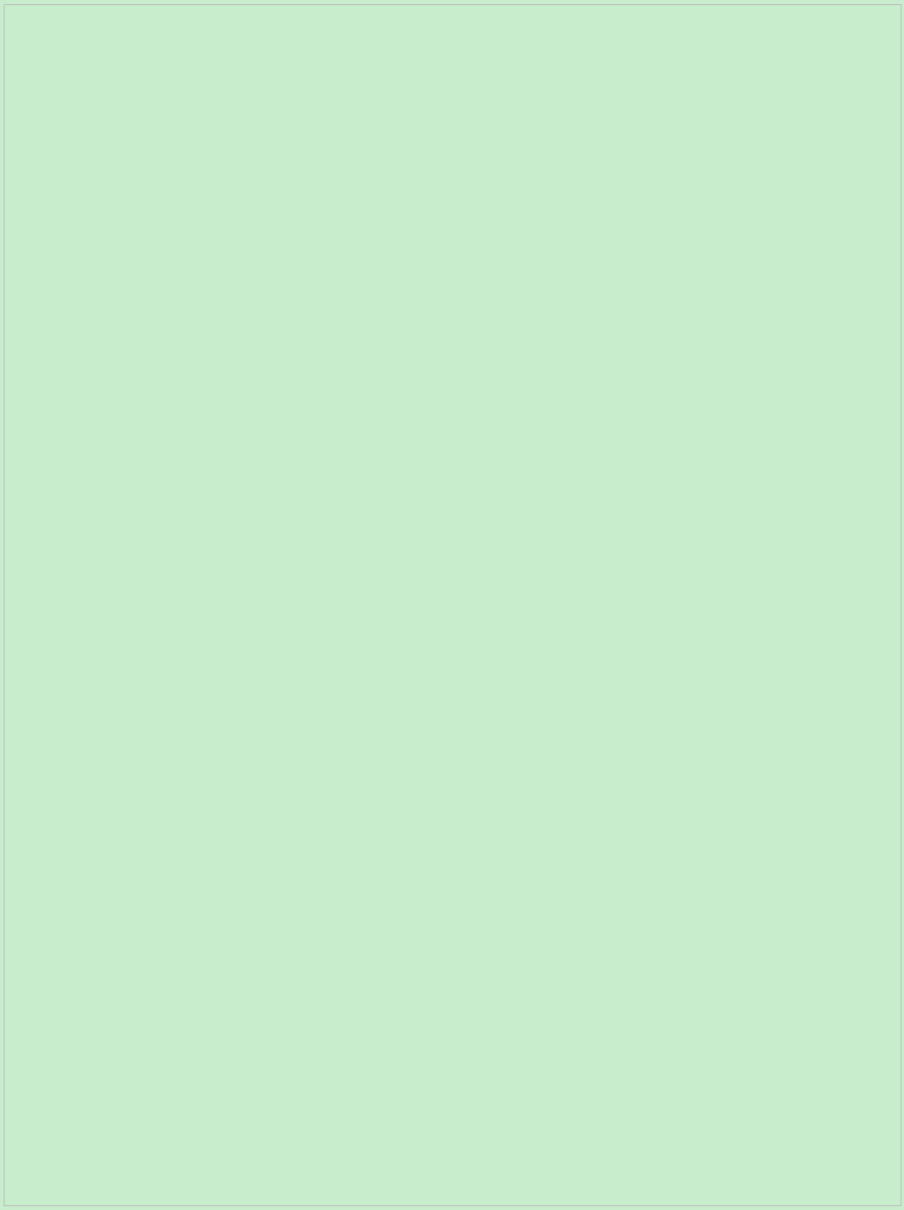


Most Lazy SKLer （Most lazy SKIer ）

Best Use of 1 Infinite Loop （Best use of 1 Infinite Loop ）

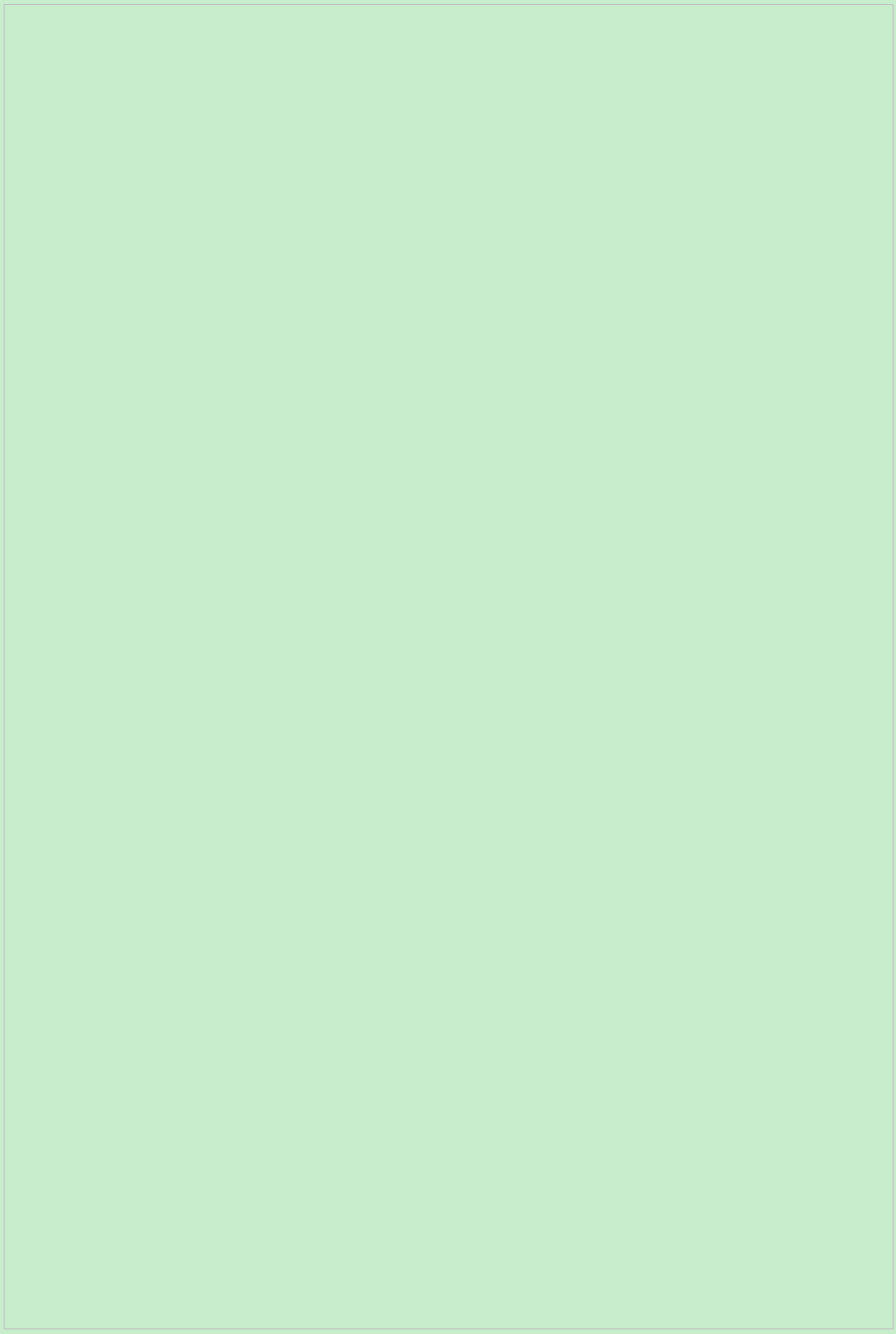
下面是以前一些比较有趣的代码：

2012，**Most Elementray Use of C**（Most elementary use of C ）



2012，**Most Functional** （Most functional ）

1998，**Best of Show** （Carl Banks' Blog: IOCCC Flight Simulator ）



==========

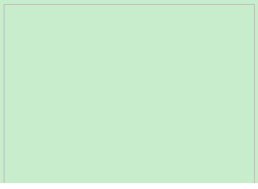
5. 数学也可以画图：用公式绘制任意图形

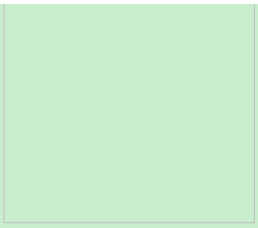
还记得一直在电视上播放的百岁山广告吗？很多人都说看不懂那个广告。其实那个广告参考了1650年著名数学家笛卡尔与瑞典公主克莉丝汀相恋的故事。具体的故事我就不再赘述了。最后，笛卡尔写给克莉丝汀的情书中出现了这样的公式：

这个公式对应的几何图形为心形，是著名的“心形线”。笛卡尔用这样一种方式，通过数学表达了对克莉丝汀的深深爱意。这封情书最后也被收录到欧洲笛卡尔博物馆中。

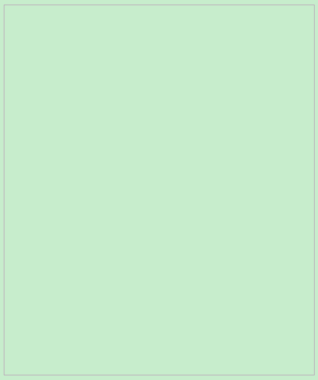
浪漫的故事到此为止，这引发了一个有趣的问题：我们能不能用数学公式，来构造任意几何图形呢？这个问题实际上是有解的。有人真的研究了这个问题，并给出了肯定的答案。链接如下： blog.wolfram.com/2013/0 ... 。我们来看一看一些有趣的图形吧~具体的生成算法课参考上面给出的链接。

美国队长曲线（Captain America-like curve）

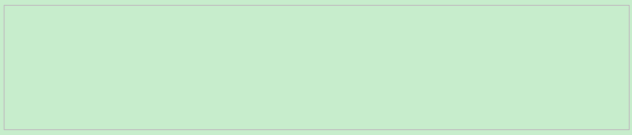




台灯曲线（Luxo Jr.like curve）



a^2 + b^2 = c^2



牛顿曲线

==========

以上。

编辑于 2015-05-31　　11 条评论 • 作者保留权利

g1n0st

wicked代码、Mr.Fisher、冯小珂 等人赞同

```
long long ago;
using ak47;
double kill;
```

发布于 2015-06-27　　11 条评论 • 作者保留权利

浪矢杂货店，打字员

琪琪de夏天、唐欣豪、汤杰 等人赞同

花费一天一夜写了一个巨复杂的代码，测试我同时发微信的三个女生跟我交往的可能性。

里面涵盖了我苦心计算的各种友好度估分，回复微信长度，回复等待时间，有没有图片，语音，女孩主动程度，
每个都有分值。
以及我发送微信过去，预计收到的微信长度，等待时间。

我还导出微信，整理了一个我的经典套话100句。
供我没话题说的时候随机选择。
经过我一礼拜的实验修正，使得女生回复我的时间，内容长度误差在10%以内。
还可以导出一堆EXCEL图表供我分析。

我每天晚上乐此不疲的和她们三个聊天做实验修改我的参数。

里面一个女孩的对我友好度逐渐远超其他女孩。
超过我设计的100分上限。果断约出来强行牵手
（虽然是程序员也知道表白死，坚决不表白，直接拉手）
最后追到这个小姑娘，完全符合我的预算。

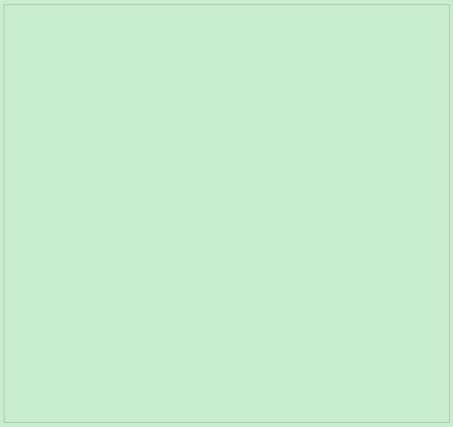
哥那段代码绝对壮绝古今。
我认为我那段代码的最终版已经把追女孩上升到一个模式化。
完全可预计结果。我简直无敌了。

后面因追到女生后一高兴删除了，后面分手懒得再写了。
分手原因是我加班太多。她家人喜欢公务员。。。

编辑于 2015-05-13　359 条评论 • 作者保留权利

段晓晨，匿几天

陈业明、闲圣、三毛 等人赞同



//其实楼上好多代码和算法我都看不懂……滚去学习了。
//但是这个答案里的东西拿来装（哔——）真的很棒。
~~这些拿来唬人真的挺方便的…尤其是那些不懂编程的…反正我当年不懂编程的时候一直觉得这是外星人的黑科技…~~

这个如何？看了你就明白。但愿知乎没把图片压坏。
显然知乎把图片压缩了……并不能获得图片里藏着的东西……容我把图片上传到网盘去。

warez.jpg_免费高速下载

下载图片，修改后缀为rar，自己查看里面的内容。看了以后你们会回来点赞的吧……

大一第一次看的时候真的是被惊艳到啊……感兴趣的可以自己搜索Warez

> Warez出品的精品动画
> 不收藏是你的错！最经典力作！！近25万倍的压缩的精品！！！
> 每年，世界各地著名的Warez组织都会推出一个小的动画片来比较，仅仅是用来炫耀其实力。64K的3D动画。要知道，一首普通的MP3，通常就有4000K左右，一张普通的JPG压缩图片也要30－40K，而这个仅有63K的3D动画，你花半个小时也看不完而且不重复带音乐.
> 推荐配置：2G/512M/GF4Ti或ATi9600、128M以上。
> prophecy《彗星撞地球》2000年时的最经典力作！相信大多网友对这个演示是相当熟悉了将1.9G的数据压缩为64K，其3D渲染和声效却令人震撼，尤其是已64K的大小竟然演示了近30分钟的不重复3D影片，其技术令人震惊～因为，事实上，这个动画的真正容量超过15G，也就是说这个Warez组织把它压缩了25万倍。注：系统必须安装有directx8.0才行。因为动画支持DX8.0加速。
> 按A键可缩小，按S键放大，按F键快速放，按R键重放……

图片…内容本身跟压缩包没有任何关系了…你可以随便换个什么，就是把两个东西放在一起糊弄人的。
搜索一下这个 copy /b 输入图片1.jpg+输入文件2.rar 输出图片.jpg 你瞬间就掌握了一门装逼技能…

当年还有小伙伴写成了界面程序…找不到了。

对了还有这个…

把下面的代码复制保存进 1.txt，然后在DOS环境（xp win7用cmd，win8 64位不支持）中转到存储txt的位置，输入debug>1.txt

```
e100 33 f6 bf 0 20 b5 10 f3 a5 8c c8 5 0 2 50 68 13 1 cb e 1f be a1 1 bf 0 1
e11b 6 57 b8 11 1 bb 21 13 89 7 4b 4b 48 79 f9 ad 86 e0 8b c8 bd ff ff e8 20
e134 0 3d 0 1 74 1a 7f 3 aa eb f3 2d ff 0 50 e8 f 0 5a f7 d8 8b d8 26 8a 1 aa
e14f 4a 75 f9 eb de cb 57 bb 21 13 8b c1 40 f7 27 f7 f5 8b fb ba 11 1 4f 4f 4a
e168 39 5 7f f9 52 8b c5 f7 25 f7 37 2b c8 95 f7 65 2 f7 37 95 2b e8 fe e fe
e181 10 79 6 c6 6 fe 10 7 46 d0 14 d1 d1 d1 e5 79 ec 5a b8 11 1 ff 7 4b 4b 48
e19b 3b d0 75 f7 5f c3 83 f7 83 a6 5d 59 82 cd b2 8 42 46 9 57 a9 c5 ca aa 1b
e1b4 4f 52 b4 92 3f ab 6e 9e a8 1d c6 3 fc e 6a e7 ae bb 5f 7b 10 b8 b4 f7 8
e1cd e2 bf 36 4e 39 9d 79 29 3f a f9 36 52 16 fb 5 e8 e5 a6 c2 e9 b0 43 d3 a3
e1e6 cf d3 fd fd cb d1 4c 5e e0 63 58 86 bb 3e 9 c1 20 bc cc 91 a3 47 81 70 b3
e1ff d6 1a 9e c2 c9 12 e7 4e ad f4 5f e3 30 e9 9 39 d7 e8 f9 f4 d2 44 e8 d7 22
e218 be e2 ce 88 25 cf 30 4a a8 29 ae 3f 47 c6 2d 85 e9 73 54 13 b e6 e0 34 65
e231 e2 50 8a 89 18 5f ce 70 99 3 5f 42 bf eb 7 ae d0 ca 5 22 8d 22 a5 b7 f0
e24a 90 81 bc 7a bc dc 5 db c0 6a 2 e5 57 38 be 60 cb ac ba a5 3b 9d f1 77 38
e263 a6 84 d1 3c af 49 d8 6a 45 a2 76 60 21 12 c0 c2 44 f2 5e bb e5 37 a9 2b
e27b ec 4a 8c 4c f2 f7 a9 58 71 2b ba 6d d6 6a e5 60 46 e0 da e5 b9 90 e5 a3
e293 f7 7f 31 60 58 f0 c4 88 10 4e 3c a3 ee 4e 11 55 8f a 92 eb db ad 7a 9c f
e2ac db 5a 28 96 da 87 ae 91 91 2d e3 5e ea df 6 95 71 67 71 40 ce d1 2e 31 6d
e2c5 c1 9c d8 6a 76 9b 4a e8 36 44 d6 76 d 30 5 ff d4 1b ac 1f 32 65 31 bf 55
e2de 26 b a4 55 e1 5d 5e 16 ed 97 48 6c 77 fb 81 86 e f9 18 bd d4 f4 8b de 1d
e2f7 ba d 47 75 3 89 4b 3e dc 27 86 1c d0 17 89 48 d1 a6 8d d4 2b 54 4e 8f b0
e310 2 e1 6b 1a 75 78 ea 21 91 13 c0 cf 78 a0 ab f3 35 c6 b4 c8 90 8d d7 45 e7
e329 c 5b a4 ba 52 10 64 f5 4a 50 b7 ec 46 22 15 23 84 30 81 5c df 61 5a 8f 67
e342 c4 63 57 6d f7 26 92 a3 1f e5 3 a5 0 54 41 8 48 7c 26 90 33 82 9c 91 b0
e35b ab 78 5d df 99 e0 b9 fc 5 36 ac d9 49 91 ab 20 a2 63 48 89 ce 5c 60 64 f0
e374 63 d9 a8 38 3b d3 e6 4c 8c 23 34 4e 20 51 93 5e 6d b4 7a 22 9b 4c f2 d3
e38c c4 f8 3 6f 47 40 f4 f8 45 9b 83 f3 83 6 31 d0 0 17 82 83 dc 67 f9 62 77
e3a5 90 3b d9 ec f3 55 96 b8 d9 db 79 55 f1 e5 8c 5e f2 e5 2e b0 b 6e e2 81 25
e3be 93 8e b5 dd 5b 46 f9 af ed 6 12 cf c9 1d f0 f7 3b 16 2d c6 58 73 8d e9 5f
e3d7 fd 5a b6 a1 94 4d 1a 8 ff eb b7 6 80 c7 86 83 b6 b9 fd 1c e0 c c3 2e a0
e3f0 2f b 3e 3 6b 29 e1 27 85 1c ea 6d df b3 a3 ed 65 4a 9a 59 3b 54 e 4b ae
e409 9e 27 f0 4d 3b c 4c 46 b7 e5 57 1b 1f 1f bb 80 86 f5 b7 ef 73 52 bf 2c c7
e422 ed a b7 81 2 f3 90 3e ee cc 6c eb f 38 1 6c 68 b1 d 45 78 b2 f f6 83 b0
e43c c4 33 df b1 d1 91 98 1e 81 a5 e2 59 9f f4 8c b6 72 8 a7 8c f6 e a3 b2 1f
e455 d9 d3 23 f0 7c 5e 5f 68 61 8b 45 da 1d 91 ec 8d 4e ea 1a 38 85 94 aa ac
e46d f2 4 f6 c4 e5 92 8e 9a 4e 83 e1 73 e8 cf 2a 5c 2b 7e f1 30 2 8a e6 28 1a
e486 3b ce bc 96 aa 7f eb 87 cd 8b 96 2d 9 59 7a a0 1a 43 62 9a 9e 4f ff 8e d9
e49f ce d6 a4 70 79 cd 65 fa 2e 92 14 29 f7 6c 74 4b 49 60 80 bb ff 41 bb 2d
e4b7 60 33 3f 98 77 9a 1 ee a6 a3 da bc ba e9 f3 72 f4 7c c3 59 2 a6 44 a4 c8
e4d0 c8 54 93 ce bd 69 bb b9 43 21 2c c4 ea 4a 5c 3f 75 60 f2 b4 91 ca 9 82 e3
e4e9 a e9 a6 20 b9 76 50 ed 47 e9 fe 6d 41 34 13 2f 28 2f 4e f4 da e 3c 78 6c
e502 b1 79 87 45 98 a4 d4 c3 b3 29 c2 4a 8b ed a6 54 e2 1b 31 62 60 ff 2c 1d
e51a 21 0 15 b2 4e 5c c 2 d 83 fa a2 f3 8a 5 12 72 4a c7 44 7c 91 d4 be b a f2
e535 70 52 fb b4 a2 df 89 de ff c4 96 73 c9 c ed d3 c9 8e 5c dc 8e d1 3b de 8c
e54e 53 a2 8b f9 e9 91 dd d6 df 6e 74 d1 dd 34 60 8f 9e 32 7f 3b ec 79 a3 83
e566 45 78 b4 2f 1c 50 7b 7a 97 b0 9d 2d c dd 8a 26 cd 7d 8c 4c 5a 8a 4c f9 a4
e57f 11 f9 2c 6c 92 e9 b5 cb 56 89 8c be f6 64 fa 25 43 fa 6f e2 c8 3a 18 a8
e597 f0 e9 f4 c2 86 e6 2b 44 67 4a b9 34 9 ed 5f 33 42 62 d4 8a 1e 5b 31 67 cd
e5b0 3d 71 6d 83 fd 36 20 69 ea 1 c3 e6 e6 de 99 aa 7 11 5b 59 8a 1f 43 83 52
e5c9 ea 5d 8c 6a 69 c7 3 eb 4e 3b 88 a5 5f b1 6e 27 5f 3 5c 28 c 9b 6c c3 f8
e5e2 e5 b9 d6 11 d6 8b fa 5c 8 c7 1 eb 45 db f3 6c 9f 16 46 61 51 ed df f bb
e5fb c0 c4 1e 64 68 98 4 79 30 94 72 df d4 cd 1f 7f 72 c6 82 2e 79 47 4e 8c 4b
e614 a2 c7 e2 36 df 76 fd a4 b6 4e db 96 40 3b 8b b5 d4 85 64 c6 0 2c ad 9d 27
e62d 14 99 82 4b bc 9 fa 94 b5 db 7c 98 eb b 13 a7 b0 79 1d 7e c5 45 aa 20 49
e646 be ff 9d 64 0 5d c ec 6 5 ad f2 38 6b ed 7a d6 b2 c7 2e 6a a6 12 4b ff 55
e660 20 3b a 77 f b9 0 9d 57 4a ad ce a4 d3 ff 1 4f fb 53 54 88 f 1 ed 4b 56
e67a 15 c8 dc 28 bf f2 72 d4 10 1f 99 42 69 9e 78 e2 47 82 93 31 d0 2d be 9f
e692 93 93 9a 1b 80 c0 10 c 53 78 a0 26 2a 96 4f 74 4b 16 c7 9c 8d ad ac fb 16
e6ab 15 c6 fd c9 a4 14 48 62 47 20 c9 41 ed 61 f8 9b f8 ff ba 39 50 65 87 ee
e6c3 bd ce 95 c0 fb a5 7e d8 cd 27 fd 2c 74 3 c1 1b 89 b9 51 d5 e3 da ef 9e 6
e6dc f0 aa a9 a7 fb 87 4c 5d cd ff 65 36 8c 73 6f 9 c6 78 9a b6 77 db df 81 68
e6f5 3b b8 ae 5d e1 af d4 e6 66 8c d6 a4 83 9f 37 3c 1 dc a2 a6 57 c2 20 1b 90
e70e 75 df cd a5 62 a5 36 79 fb 35 8a 9b b0 a0 a5 c3 37 6f 80 72 bc 52 30 8d
e726 9f 7a 64 d3 7 41 45 d8 68 97 f2 aa 1c a1 6c 7c 9d 32 7d ad 15 b1 53 e3 33
e73f 8a ed e9 49 d4 cf dc 96 22 37 36 11 9d 7f f0 4d e0 62 31 b1 c7 69 c4 79
e757 ac 20 1 e8 3c 6a 8c 32 cb 52 63 36 68 f4 10 2b 9c 21 4f df 5d 60 92 39 91
e770 e2 f9 c9 7d ca 48 3 3f 21 dd 6c f 23 2e 61 3a 9f ba c3 f9 4e 7 ea ed ef
e789 71 4a 72 3a ed 23 3d 77 b5 ed d5 1d f6 a4 99 fa ef 98 dd 2 98 80 b6 7c a3
e7a2 62 96 7b 8e bf 7b 81 9f 9a ce 3f 12 40 2e 25 db 84 16 dd 2e 86 f f4 b2 7e
e7bb 5e b4 14 6a f3 29 b1 a4 57 d5 a8 17 6f 87 a4 74 5b 9b 17 79 f1 ec 33 c8
```

e7d3 f0 1d b2 7e a8 4d 95 7f 5f 9 d5 1a 5a 45 f4 41 c6 d 3f eb 66 2a c0 e8 5b
e7ec 3c bd 50 ad f1 53 9d 2e 45 9a d8 7d 2c 17 a8 6e 15 48 13 39 53 ed 3d 78
e804 ad f 3a 65 a3 3e 2e fa ca 7 94 4a 1f b4 d8 7e 47 8a 8e de e7 7e 34 c1 69
e81d 7f 6a aa 66 58 18 31 24 72 13 22 34 8a 56 36 87 df c2 d 8e 3f 71 a2 5f 25
e836 8b 8d 4 78 fd c9 45 d1 55 79 c1 9f 13 84 1b c8 5 db 95 d0 7c 64 96 20 51
e84f c4 e0 5e ee 47 8a 11 ac fb 9 e0 bb 40 db 86 84 12 93 b9 c9 f2 9c 63 47 c9
e868 eb ad 1 3e fa 6d 3f a 64 5b 58 56 27 f ca 5d e0 30 bc 3e 10 5d ec 17 28
e881 85 5 51 8e 95 a3 94 3a a8 f1 96 f2 f 29 5c 97 dc 47 db 9d 6c 63 e8 e7 f0
e89a e4 a 70 f8 f1 47 54 d3 2d 32 7c ef bb 9a b4 1b 0 2b d6 dd e7 30 b a2 75
e8b3 c7 f5 d0 31 d7 d2 8a b0 ac 1c 6d 60 3a f7 c2 db 1e 6d 7 f6 8f 35 88 e5 7f
e8cc 3c 26 81 34 a0 32 a3 25 18 6e 73 b2 a0 f1 cb 86 61 e7 65 8b 76 98 19 6f
e8e4 c0 62 9b a3 cc 18 5e 40 12 97 2b d0 15 79 de 19 ea df 7a 59 2f b5 d7 39
e8fc 52 e2 6 f1 3 a0 a5 d9 1b 88 93 4d 30 c8 2d f5 db 55 ea 85 6f a 3f dc bd
e915 57 15 6a a3 a3 3e 8e ad 2d da a0 ca 75 7c 57 8b c5 cb b 1d 2c 8e c6 96 2e
e92e 6d 59 83 7d 64 72 ca 80 2e 6 a4 ff f6 f2 d5 1e 7 4 ba 34 6e 9 86 25 aa 4e
e948 e0 7f f5 32 47 3e 7c 43 d8 28 c4 1c 11 1d bd 33 3 b5 ca 13 43 34 2 b1 a0
e961 57 ed 9d 3c 23 d4 45 b2 6e 81 6e af 3e 67 90 be 59 a5 45 34 53 46 85 d1
e979 25 ee 7d cb a4 db 12 c3 aa 17 61 9a fb 66 40 76 fe 3a 69 96 c0 91 14 a7
e991 5d cc 9f f6 73 59 ee b8 55 97 20 26 ff 99 ec 72 41 b5 27 21 6e ae 8a d0
e9a9 e4 d3 da 6f c4 53 c5 f8 b3 a7 a1 5d 66 93 d8 b1 89 40 23 92 c0 90 fb cb
e9c1 e7 6b 4e 51 0 5d 57 f7 cd 1 e2 88 bf 44 9f ef c4 33 ce fa 46 46 a1 86 b
e9da 7a 84 66 66 b9 2 ec 10 c6 a1 d4 c1 18 33 b1 d1 2 18 ad 2f 53 e4 b9 33 59
e9f3 be 3c af 80 4c 8a d5 76 c 3b a7 e2 97 94 15 75 4d 17 d5 97 cf f9 4a d0 6e
ea0c bb 27 20 fc f1 f5 9 a8 df 4d b6 5d f0 1d 69 3b 76 35 82 a4 f3 56 64 39 5b
ea25 6b b3 7 e7 5 8e 82 11 22 a8 1a db c8 3e 67 4a 3 7e 72 51 d6 3d 1a 1c f6
ea3e b8 da 4b 18 8a 15 9d d0 a4 84 96 3e cd 3 f9 3a 30 f3 fb 8f 6e 2 73 eb 52
ea57 93 95 cf dc 6f 48 fb ab d2 a9 70 b4 e2 23 8d 72 86 a8 fa 78 98 1d c5 fe
ea6f 8a 51 88 2b b7 58 b0 ca ae 40 8a 33 32 75 1 6 c0 d4 b7 da 2a a7 bb ad f7
ea88 48 98 5a bc d3 d1 e6 16 97 c3 80 ab 73 ac 32 11 41 1f d 5d aa 0 dc d9 6e
eaa1 fc 30 6 ef 11 60 27 a2 5f eb 5f b9 35 8 23 4 be 10 c0 85 3e 55 b3 82 fd
eaba f7 c3 24 9f 2d 83 94 32 36 de ff 7c 87 7f 4a 80 7 2 23 cf a4 52 eb 3e 19
ead3 a0 b4 a 94 1a 40 58 d9 16 6d c0 64 c4 69 ed 60 46 65 cb df 58 38 0 51 c3
eaec ad a0 37 e4 cf ab f7 6c 24 7d 9 48 65 4a 9f 91 ad 1c 79 a4 a1 78 55 c e8
eb05 44 5b d ef 51 bd ea 2d a7 42 57 ab 3a 4f 2 b 3 19 6a 4d 72 76 5c 97 0 6c
eb1f c5 5d bc dd e7 81 cf 8d 34 38 50 3c 98 58 cc 41 aa 99 90 af fe 4e 96 77
eb37 ed 54 18 ce 2c d1 5d 34 cb 79 50 ff 28 96 44 e0 51 64 6 a8 b7 6e 8c 62 c4
eb50 66 95 81 4f 8c f6 26 ba ea 5d d2 79 b1 e4 e9 29 fc a fd b3 85 8c e6 52 dd
eb69 33 bd 5d c7 39 ef 6 ef 9e a6 6a 61 9c 9f d5 54 b4 fa a1 d4 10 9b ff 7e 33
eb82 11 52 99 c7 26 6e a1 36 8a ad ee 48 7a 2c 7f d5 b7 27 8a 6b 37 c 71 39 85
eb9b 9c ba a8 a 17 b9 d0 51 56 95 c2 3b 5 a7 31 c5 8b 5c 95 6e 4c 89 6f 17 ef
ebb4 d4 5a a 77 65 e1 49 b2 e8 72 ac 3c f0 6b 71 fa 3 c7 ca fc ad f9 55 22 ec
ebcd 58 2f 1c fa 29 cf 73 b4 ad 51 5c f8 66 70 59 5d 70 3e d1 3f c4 eb ec f1
ebe5 7 78 6a 93 67 9f 44 fc cb 5b 95 ff 74 c0 b7 42 77 26 c9 aa 8c ed 39 a2 db
ebfe 9c b3 eb 3d 4a 1e 9b 89 e4 d8 a8 27 74 ef a3 ed a5 24 5d bb ab d0 fe a1
ec16 29 ab df 75 a a6 23 0 cc f1 14 72 9b 1a 55 7e e5 d1 da 98 dc c4 cf ab 34
ec2f ba 8d de 4a 59 6 13 dd d8 44 3c e bb 56 95 ae 97 e2 3b 49 e5 9a 6b a2 53
ec48 c1 33 35 24 1b 33 17 c3 8a 8c 12 3d 3d 4e 5b 75 22 30 67 4f a0 5d 3a 78
ec60 88 a 11 35 7 b1 77 42 32 a8 c3 bb 20 fb 98 5 d6 ac e7 3a 63 35 90 93 9e
ec79 44 24 2e 1b d7 8c aa 29 53 4d d9 ab eb e6 1 56 c4 fd 54 a3 bd 14 5b b0 8f
ec92 ce be 23 24 93 c4 48 18 a3 e7 4 5 4b 78 cc 79 dd 3 56 a4 ed dd 5f 98 41
ecab 1b 68 4c c1 bb 41 c2 1e 3e 94 8e ef 28 1e b 76 e 4f 36 b1 c 6e e2 18 17
ecc4 20 fc 35 40 1f e4 6d a4 18 bb bc d5 9e ea 85 86 af af 63 d4 13 66 92 c4
ecdc 2b 69 84 ca 23 2b d3 66 81 6b 81 73 26 4 85 36 21 4c 49 44 75 64 39 16 3c
ecf5 ed e0 6d 44 75 45 30 43 68 c0 78 fc d0 17 b eb 81 3e c3 ba 1b f 4d ae c5
ed0e 55 1f c 39 12 5d 8 65 f1 34 59 de dd 98 56 17 43 38 66 49 9a eb db c1 87
ed27 51 38 cc b7 5f 98 fd 43 be 2d bb 74 f3 f8 f2 36 3d a4 34 a5 7e d2 26 cc
ed3f 84 1f ea 56 f0 80 18 69 4d 88 41 fc 56 fd 41 3b 1e e 9 27 4f f6 3b 62 4e
ed58 5a 1b 2a 4e 85 8c b2 4f 79 ef 59 4e e 73 3d bd c4 ca 60 e7 4a 47 90 b5 8
ed71 2a f0 4e dc ba 66 ae 48 2b 31 73 a2 11 c 32 ff 54 14 77 6b d6 58 4b bf ee
ed8a f6 6a bc dd 1 88 d da a9 f 81 24 c5 f8 72 9a db d5 c8 2a 80 a9 16 d7 c6
eda3 b1 91 c0 a9 95 40 b5 b3 a8 2a 28 c6 92 16 ab 54 7d f8 93 5f 3a 17 c8 45
edbb a9 f0 e0 71 23 76 53 38 a5 a1 cc d4 f1 f2 3c 2b 46 43 a1 d5 ba e d7 19 7a
edd4 c2 e1 8f 67 1d d 98 9d a1 79 9d 1b 20 7f 4d e7 bf f9 ff fe aa 28 ab 8f c
eded 4d 50 33 e3 26 fc 3c 3 3a 2b 26 12 f7 1 8f ee 97 4c e6 6 2b d9 1f a1 4a
ee06 77 44 d4 8b b7 3e 5e 2d 18 c3 54 68 99 a8 8d 92 96 9e 9d ab 33 38 ff b8
ee1e ee 78 c6 7b b5 84 95 d3 6 27 ae 5d 27 38 a 38 8e f0 1 a5 96 4b d7 9b 42
ee37 e5 6f 57 75 4c e9 78 2d 5b ec b6 d2 29 e2 a8 92 95 9c 65 2a 3e bf 8d e0
ee4f bf b3 ac c8 e 7e 13 af 88 26 7d 48 5a c7 39 29 36 d2 90 e8 3b 3 d0 61 1a
ee68 d2 e8 a8 f ba 8e a1 9f df 12 ab 54 7 23 98 de 62 af 4c 7e d4 fb 6b 2 6e
ee81 40 40 37 b7 73 f2 d8 81 be 29 d2 99 c0 73 25 1a 3c 92 75 6e bd d7 79 79
ee99 4 14 c0 4e 99 57 66 93 74 ec b0 29 7c df 61 b0 3 3a d1 c3 fa a4 f7 f 9f
eeb2 d3 f 0 b9 2a 5a 3a c5 88 25 b8 b9 cc 82 3 57 3a e1 7b 51 75 70 a6 74 1a
eecb ca cb 3 18 68 ca 77 fe 1b ad cd 68 7f 36 85 fc b7 4f a0 11 da 69 fa 79 87
eee4 d6 b9 21 dd 3e 70 db dc 84 d4 6e d1 20 4 af f6 32 a2 8e d 54 25 fe 7 54
eefd e 7a 74 4b a0 4b f7 f4 e8 74 22 e9 98 70 fb 25 2e f4 64 57 75 28 85 45 53
ef16 3a 2e e2 3c 54 36 e9 29 6 67 59 43 10 7e c1 49 cd 5e f9 97 a 58 5f 8a 11

```
ef2f 4f 3d 9a e2 2b 22 58 fa be fc 69 91 7a 8c 3f 77 9f c9 3b 54 26 23 93 b3
ef47 85 de ae f5 bd c5 47 4c c4 cd 5e ad bc 8f ba 31 f6 e4 70 fb 6e a7 96 d5
ef5f ad 10 80 39 43 97 4f 10 cc 1b 8f 8d cd 4c 63 4 d8 1e 85 70 41 6c a8 eb df
ef78 7f 36 c5 60 a7 12 9 16 73 fe 75 3a 2d 40 29 7d aa a 5c 2 29 23 0 a6 e5 6b
ef92 24 6d 9b 20 e5 7 cb 40 b0 38 59 9c a7 69 6a 70 d3 38 ef e2 b2 11 3e ea 2a
efab f9 2b 2e 43 1d 65 cf d6 1b ef 83 5a 5f e6 c5 62 16 ca 5e 4c a6 39 e4 53
efc3 2d 23 d2 5e 7e 15 54 8a 8 b7 3d bb 88 59 b9 9e a2 7c 42 1f a2 77 3c 5b 9
efdc 6d fa 8f 21 46 1a 3e ed ce 49 56 1d 29 2d 70 3 a7 6f 75 ac 1 87 ff 27 86
eff5 73 49 28 85 2d 97 7a 84 e 37 3d 86 10 21 4c e2 74 62 6b 51 70 8f 15 72 f3
e100e 81 b2 a9 9d 8a 63 ad 1b d5 aa 8a dc 96 3c e7 47 16 51 fc 87 50 9 b7 60
e1026 29 33 52 fb b0 df 70 c5 65 4a 60 3b c d7 a8 29 47 51 f7 8a 77 f3 99 3f
e103e 38 16 60 de 68 27 b2 24 7 62 a2 fd 40 86 b2 75 c3 3c 2f 3d fa 9 d9 a9 9a
e1057 71 3c ce 46 94 0 f9 bc 46 7f b8 2e 85 7f 7d d3 8d ea b4 63 81 59 10 bb
e106f 57 d0 b6 ab e1 83 74 1e 25 d5 73 78 18 b1 60 62 c f4 76 8d 17 d5 ed 23
e1087 23 e4 f6 32 64 5a 61 9 63 f6 92 57 d5 29 40 d6 3b ba 63 72 18 0 25 1b 7
e10a0 ee 7f 25 4a fa 6 74 19 46 e3 e8 89 7a c6 56 54 a7 43 13 4e bf 97 a5 6f
e10b8 99 2f ac 33 4d fa 58 3a 5a a a4 1a 74 62 c8 4f 3b 78 9 d7 ee 7e ee 2d 69
e10d1 30 40 ea 47 82 3b 85 8e 3 23 8f 74 4e 8 35 ab 74 4 1 57 d5 85 b1 6b 1e
e10ea f4 7d 1e d2 1e b3 fe f3 12 10 32 39 51 48 2d 6f e5 d3 a3 8c 8 8

g
rcx
fff
n1.com
w
q
```

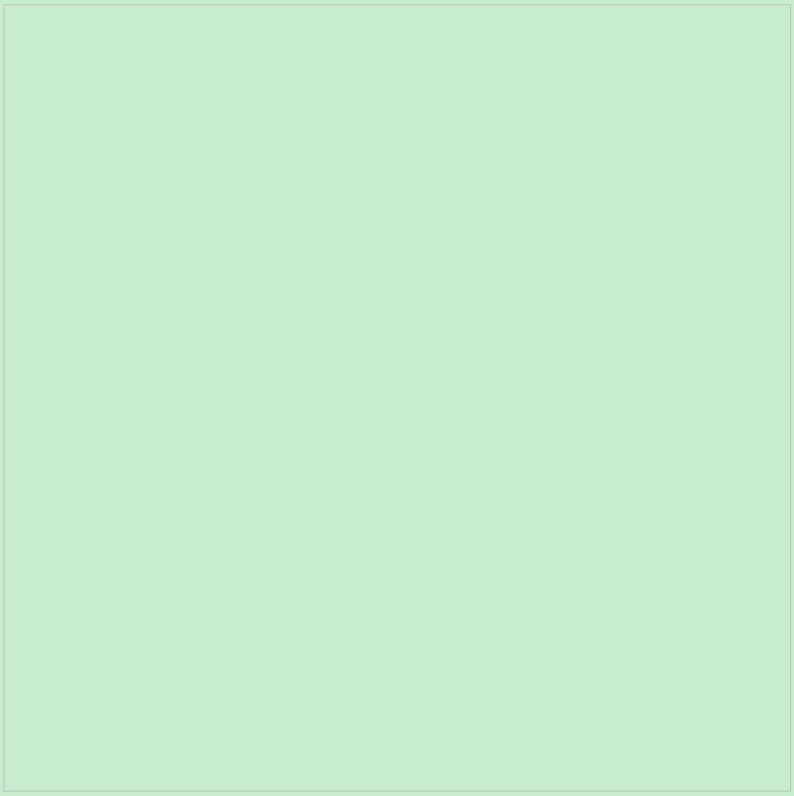
编辑于 2015-05-31　　14 条评论 • 作者保留权利

朴三世，程序猿

SteveLeeLX、Snowstar、陈天文 等人赞同

mame/quine-relay • GitHub

突然发现已经100种语言了，想看50种语言的进50分支。



璀璨无比谈不上，反正让我震惊了会儿，原来可以这样玩

编辑于 2015-05-15　　9 条评论 • 作者保留权利

韦易笑，二十年老程序员

陈晟祺、王文涛、一大截 等人赞同

24行内实现 BASIC语言（老外写的）：

```
#define O(b,f,u,s,c,a)b(){int o=f();switch(*p++){X u:_ o s b();X c:_ o a b();default:p--;_ o;}}
#define t(e,d,_,C)X e:f=fopen(B+d,_);C;fclose(f)
#define U(y,z)while(p=Q(s,y))*p++=z,*p=' '
#define N for(i=0;i<11*R;i++)m[i]&&
#define I "%d %s\n",i,m[i]
#define X ;break;case
#define _ return
#define R 999
typedef char*A;int*C,E[R],L[R],M[R],P[R],l,i,j;char B[R],F[2];A m[12*R],malloc
(),p,q,x,y,z,s,d,f,fopen();A Q(s,o)A s,o;{for(x=s;*x;x++){for(y=x,z=o;*z&&*y==
*z;y++)z++;if(z>o&&!*z)_ x;}_ 0;}main(){m[11*R]="E";while(puts("Ok"),gets(B)
)switch(*B){X'R':C=E;l=1;for(i=0;i<R;P[i++]=0);while(l){while(!(s=m[l]))l++;if
(!Q(s,"\""))U("<>",'#');U("<=",'$');U(">=",'!');}d=B;while(*F=*s){*s=='"'&&j
++;if(j&1||!Q(" \t",F))*d++=*s;s++;}*d--=j=0;if(B[1]!='=')switch(*B){X'E':l=-1
X'R':B[2]!='M'&&(l=*--C)X'I':B[1]=='N'?gets(p=B),P[*d]=S():(*(q=Q(B,"TH"))=0,p
=B+2,S()&&(p=q+4,l=S()-1))X'P':B[5]=='"'?*d=0,puts(B+6):(p=B+5,printf("%d\n",S
()))X'G':p=B+4,B[2]=='S'&&(*C++=l,p++),l=S()-1 X'F':*(q=Q(B,"TO"))=0;p=B+5;P[i
=B[3]]=S();p=q+2;M[i]=S();L[i]=l X'N':++P[*d]<=M[*d]&&(l=L[*d]);}else p=B+2,P[
*B]=S();l++;}X'L':N printf(I)X'N':N free(m[i]),m[i]=0 X'B':_ 0 t('S',5,"w",N
fprintf(f,I))t('O',4,"r",while(fgets(B,R,f))(*Q(B,"\n")=0,G()))X 0:default:G()
;}_ 0;}G(){l=atoi(B);m[l]&&free(m[l]);(p=Q(B," "))?strcpy(m[l]=malloc(strlen(p
)),p+1):(m[l]=0,0);}O(S,J,'=',==,'#',!=)O(J,K,'<',<,'>',>)O(K,V,'$',<=,'!',>=)
```

```
O(V,W,'+',+,'-',-)O(W,Y,'*',*,'/',/)Y(){int o;_*p=='-'?p++,-Y():*p>='0'&&*p<=
'9'?strtol(p,&p,0):*p=='('?p++,o=S(),p++,o:P[*p++];}
```

运行后输入程序（记住字母大写）：

```
10 A=1
20 B=1
30 FOR I=1 TO 10 DO
40 C=A+B
50 PRINT A
60 A=B
70 B=C
80 NEXT I
90 END
```

然后输入命令：

```
RUN
```

即可求解斐波拉契数列。

再来一段我写的空战游戏：

WINXP下（或者DOSBOX：D-Fend Reloaded ）在DOS窗口中运行DEBUG，然后把横线下的内容复制、粘贴到DEBUG窗口中，回车就可以见到了。

```
-------------------------------------------------------------------
e100 e8 5f 0 e8 4c 1 e8 ed 0 b8 0 4c cd 20 f f 0 8 7 1 7 1 6 7 7 1 0 f 1 0 0
e11f 0 fe c2 cb c4 c1 8a d3 c5 df 8a cc c5 d8 8a da c6 cb d3 c3 c4 cd 86 8a
e137 c8 d3 8a d9 c1 d3 dd c3 c4 ce 99 9a 9a 9a ea c2 c5 de c7 cb c3 c6 84 c9
e14f c5 c7 a7 a0 0 87 97 8a ef f2 ec e5 f8 e9 ef 8a 97 87 0 66 60 b8 0 11 a3
e168 e 1 5 80 bb a3 16 1 5 0 1 a3 10 1 5 80 0 a3 12 1 5 10 2 a3 14 1 8b 3e 10
e185 1 fc b9 80 2 33 c0 f3 aa 1e 33 c0 8e d8 8e c0 be 24 0 66 ad 1f 66 a3 18
e19e 1 8c c8 66 c1 e0 10 b8 1d 2 bf 24 0 fa 66 ab b0 34 e6 43 b8 87 0 e6 40
e1b7 8a c4 e6 40 fb 66 33 c0 1e 7 b8 13 0 cd 10 be 7e 3 bb 0 0 8b 3e 16 1 ac
e1d1 3c c0 73 4 aa 43 eb 9 24 3f 50 ac 59 f3 aa 43 43 80 fb 9b 72 e9 33 c0 cd
e1ea 1a 81 e2 ff 7f 89 16 1c 1 66 61 c3 33 c0 8e c0 66 a1 18 1 bf 24 0 fa 66
e203 ab b0 34 e6 43 33 c0 e6 40 e6 40 fb 66 33 c0 b8 3 0 cd 10 1e 7 e8 4d 0
e21c c3 60 1e 6 8c c8 8e d8 8e c0 33 c0 e4 60 8b c8 83 e1 7f 8b 1e 10 1 3 d9
e235 24 80 f6 d0 c1 e8 7 88 7 e4 61 c 80 e6 61 24 7f e6 61 b0 20 e6 20 7 1f
e24e 61 cf c3 c3 60 b4 2 b7 0 ba d 0 cd 10 b3 3 be 54 1 e8 9 1 61 e8 b1 1 c3
e269 be 20 1 e8 fe 0 33 c0 cd 16 c3 60 8b 36 e 1 bf 40 1f b9 c0 5d fc b8 0 a0
e283 8e c0 f3 a5 e 7 61 c3 60 8b 3e e 1 b9 c0 5d e 7 33 c0 fc f3 ab 61 c3 c8
e29d 0 0 0 60 8b 4e 4 8b 56 6 8a 46 8 81 f9 40 1 73 1a 81 fa 96 0 73 14 8b 3e
e2b8 e 1 8b da c1 e2 8 c1 e3 6 3 da 3 d9 3 fb 88 5 61 c9 c3 c8 2 0 0 60 8b 1e
e2d4 16 1 ba 0 0 83 6e 4 8 83 6e 6 8 c7 46 fe f 0 b9 0 0 33 c0 8a 7 43 3c 0
e2f0 74 1e 50 8b 46 6 83 7e 8 0 74 4 3 c2 eb 3 3 46 fe 50 8b 46 4 3 c1 50 e8
e30b 8f ff 83 c4 6 41 83 f9 10 72 d3 ff 4e fe 42 83 fa 10 72 c7 8b 4e 4 8b 56
e324 6 61 c9 c3 c8 0 0 0 52 66 a1 1c 1 66 69 c0 35 4e 5a 1 66 40 66 a3 1c 1
e33e 66 c1 e8 10 66 25 ff 7f 0 0 99 f7 7e 4 8b c2 5a c9 c3 c8 0 0 0 8b 1e 12
e358 1 b9 20 0 b8 0 0 83 3f 0 74 7 83 c3 10 40 49 75 f4 c9 c3 fc ac 3c 0 74 a
e373 b4 e 34 aa 60 cd 10 61 eb f1 c3 c3 0 1d 19 c4 0 19 1d c1 c4 c7 0 c2 19
e38c c4 0 19 0 c1 c4 c4 0 c2 70 0 19 c1 c4 c2 28 1d c2 70 28 70 0 c3 70 c2 28
e3a6 c2 70 c2 28 c2 19 70 c2 28 c6 70 28 70 19 c2 28 c3 70 c2 28 19 28 c2 70
e3be c2 0 c2 70 19 c3 28 70 c3 28 19 28 70 c4 0 70 19 28 70 28 c3 70 28 19 70
e3d7 c6 0 1d 70 0 28 0 70 0 70 1d c7 0 1d 70 0 28 36 70 0 70 1d c7 0 1d c2 0
e3f2 28 36 70 c2 0 1d c9 0 28 70 19 c2 70 cb 0 28 70 19 c2 70 cb 0 28 c4 70
e40b cc 0 28 c2 70 cd 0 28 19 cf 0 70 c8 0 c8 32 0 0 56 57 c7 46 fe 9c 2 c7
e425 46 fc cd 2 c7 46 fa 28 3 c7 46 f8 51 3 c6 46 f1 1 c6 46 f0 0 c6 46 ef 0
e43f 66 c7 46 e8 0 0 0 0 c7 46 e2 a0 0 c7 46 e0 78 0 c7 46 d2 0 0 c7 46 ce 0
e45a 0 a1 10 1 89 46 ec a1 14 1 89 46 f6 a1 12 1 89 46 f4 c7 46 de 0 0 8b 76
e474 f6 eb 38 68 40 1 ff 56 fa 59 89 4 68 c8 0 ff 56 fa 59 5 ce ff 89 44 2 83
e48e 7e de 35 7d c c7 44 4 1 0 c7 44 6 17 0 eb a c7 44 4 2 0 c7 44 6 1c 0 ff
e4aa 46 de 83 c6 8 83 7e de 50 7c c2 e9 94 3 66 8b 46 e8 66 89 46 e4 eb 14 66
e4c3 60 33 c0 cd 1a 8b c1 66 c1 e0 10 8b c2 66 89 46 e4 66 61 66 8b 46 e4 66
e4db 2b 46 e8 66 83 f8 c 72 de 66 8b 46 e4 66 89 46 e8 b8 8b 2 ff d0 c7 46 de
e4f4 50 0 8b 76 f6 eb 36 8a 44 6 50 ff 74 2 ff 34 ff 56 fe 83 c4 6 8b 44 4 1
e50e 44 2 81 7c 2 96 0 7e 14 68 40 1 ff 56 fa 59 89 4 6a 3c ff 56 fa 59 f7 d8
e528 89 44 2 ff 4e de 83 c6 8 83 7e de 0 75 c4 8b 5e ec 80 7f 4b 0 74 f 83 6e
e542 e2 2 83 7e e2 0 7d 5 c7 46 e2 0 0 8b 5e ec 80 7f 4d 0 74 10 83 46 e2 2
e55c 81 7e e2 40 1 7e 5 c7 46 e2 40 1 8b 5e ec 80 7f 48 0 74 f 83 6e e0 3 83
e576 7e e0 0 7d 5 c7 46 e0 0 0 8b 5e ec 80 7f 50 0 74 10 83 46 e0 2 81 7e e0
e590 96 0 7e 5 c7 46 e0 96 0 8b 5e ec 80 7f 1 0 74 3 e9 b0 2 8b 5e ec 80 7f
e5aa 1d 0 74 33 80 7e f0 0 75 31 c6 46 f0 1 ff 56 f8 c1 e0 4 8b 7e f4 3 f8 80
```

```
e5c4 7e ef 2 7d 1c fe 46 ef c7 5 2 0 8b 46 e2 89 45 8 8b 46 e0 5 f7 ff 89 45
e5de a eb 4 c6 46 f0 0 c7 46 de 0 0 8b 7e f4 e9 a3 1 8b 45 8 89 46 d6 8b 45 a
e5f9 89 46 d4 8b 5 89 46 d0 3d 1 0 74 b 3d 2 0 75 3 e9 a7 0 e9 6b 1 83 7d 2 0
e615 74 6d 8b 46 d6 2b 46 e2 89 46 da 83 7e da 0 7d 5 f7 d8 89 46 da 8b 46 d4
e62e 2b 46 e0 89 46 d8 83 7e d8 0 7d 5 f7 d8 89 46 d8 83 7e da d 7d a 83 7e
e647 d8 d 7d 4 c6 46 f1 0 6a 2 ff 56 fa 59 40 1 46 d4 81 7e d4 a0 0 7e 5 c7
e661 46 d0 0 0 6a 8 ff 56 fa 59 b c0 75 25 8b 46 d6 3b 46 e2 7e 5 b8 ff ff eb
e67b 3 b8 1 0 1 46 d6 eb 10 ff 45 4 8b 45 4 3d 28 0 7e 5 c7 46 d0 0 0 8b 45 2
e697 8b 55 4 83 e2 1 b c2 75 3 e9 d8 0 6a 1 ff 76 d4 ff 76 d6 ff 56 fc 83 c4
e6b1 6 e9 c7 0 8b 46 d4 5 fb ff 89 46 dc eb 27 6a 9 ff 76 dc 8b 46 d6 5 fb ff
e6cb 50 ff 56 fe 83 c4 6 6a 9 ff 76 dc 8b 46 d6 5 3 0 50 ff 56 fe 83 c4 6 ff
e6e5 46 dc 8b 46 d4 5 5 0 3b 46 dc 7f ce 83 6e d4 4 83 7e d4 ec 7d 8 c7 46 d0
e6ff 0 0 fe 4e ef c7 46 dc 0 0 8b 46 f4 89 46 f2 eb 65 8b 5e f2 83 3f 1 75 56
e719 83 7f 2 1 75 50 83 7e d0 0 74 4a 8b 46 d6 2b 47 8 89 46 da 83 7e da 0 7d
e733 5 f7 d8 89 46 da 8b 5e f2 8b 46 d4 2b 47 a 89 46 d8 83 7e d8 0 7d 5 f7
e74c d8 89 46 d8 83 7e da f 7d 19 83 7e d8 f 7d 13 8b 5e f2 c7 47 2 0 0 c7 46
e766 d0 0 0 fe 4e ef ff 46 ce ff 46 dc 83 46 f2 10 83 7e dc 20 7c 95 8b 46 d6
e77f 89 45 8 8b 46 d4 89 45 a 8b 46 d0 89 5 ff 46 de 83 c7 10 83 7e de 20 7d
e798 3 e9 54 fe 6a 14 ff 56 fa 59 b c0 75 3b ff 56 f8 c1 e0 4 8b 56 f4 3 d0
e7b1 89 56 f2 8b 5e f2 c7 7 1 0 c7 47 2 1 0 c7 47 4 0 0 68 40 1 ff 56 fa 59
e7cc 8b 5e f2 89 47 8 6a a ff 56 fa 59 5 ec ff 8b 5e f2 89 47 a b8 96 0 2b 46
e7e6 ce 89 46 d4 83 7e d4 0 7d 5 c7 46 d4 0 0 8b 46 d4 89 46 de eb 11 6a 4 ff
e800 76 de 68 3f 1 ff 56 fe 83 c4 6 ff 46 de 81 7e de 96 0 7c e8 81 7e d2 8c
e819 0 7d f c6 46 f1 1 8b 46 d2 25 1 0 89 46 da eb 5 c7 46 da 1 0 83 7e da 0
e834 74 e 6a 0 ff 76 e0 ff 76 e2 ff 56 fc 83 c4 6 b8 74 2 ff d0 ff 46 d2 80
e84d 7e f1 0 74 3 e9 63 fc 80 7e f1 0 75 2e 66 8b 46 e8 66 89 46 e4 eb 14 66
e866 60 33 c0 cd 1a 8b c1 66 c1 e0 10 8b c2 66 89 46 e4 66 61 66 8b 46 e4 66
e87e 2b 46 e8 66 3d d0 2 0 0 72 dc 5f 5e c9 c3 ff 53 4b 59 57 49 4e 44 30 35
g
------------------------------------------------------------------------
```

游戏运行于DOS环境，所以不必当心它是个病毒 ：－），用方向键控制运行，CTRL发射激光，
如果运行在WINXP下面，粘贴操作只需要点击DEBUG窗口的图标，选“编辑”即可。

游戏效果：

我 2004年实现的竖版空战射击游戏，整个代码 1K以内。

Win7下 DosBox的用法（没有 WinXP和 VmWare时）：

1. 下载安装 D-Fend Reloaded 最新版（带GUI和 FreeDOS 的DosBox） 并运行
2. 复制游戏代码（横线中内容）保存到 C:\Users\用户名\D-Fend Reloaded\VirtualHD\game.txt
3. 双击 D-Fend Reloaded窗口中的 DosBox，启动DosBox窗口
4. 按CTRL_F12将CPU Speed调到10000以上
5. 输入命令：debug < C:\game.txt，按回车启动游戏

编辑于 2015-05-14 62 条评论 • 作者保留权利

罗宸，傻X问题大家都爱答。。。

ChenYang Li、龙猫、何洋 等人赞同

Y组合子可以允许你**不使用递归语法定义递归函数**，用CoffeeScript实现一个就长这样：

```
Y = (f) -> #the Y combinator
 ((x) -> (x x)) ((x) -> (f ((y) -> ((x x) y))))
```

然后，你可以用它来定义斐波那契数列：

```
fib = Y (f) ->
 (n) ->
  if n < 2 then n else f(n-1) + f(n-2)
```

然后，你可以打印出前10个斐波那契数试试：

```
log -> list map(fib) range(10)
```

这里可以试玩哦（注意**不要打印超过30个**，因为这是指数复杂度的！）：Try Coffee

这段代码有意思的地方在于：Y和fib两个函数的定义都没有使用递归语法，却就这么自然而然地定义出了递归函数，所以你可以从中感受到两件事情：

1，递归是一件如此普遍的事情，以至于编程语言根本不用刻意实现这样的语法，它就自然而然地存在了
2，人类的智慧真是深不可测（以及FP的水好深。。。）

所以你知道为什么Paul Graham要把自己的公司叫Y Combinator了吧，现在google这个词搜到的都是他的公司，取这名字是不是很贱！

当然lambda演算里还有很多有趣的东西，Y Combinator仅仅是冰山一角 ：）

参见：
不动点组合子
Mike's World-O-Programming

编辑于 2015-05-13　6 条评论 • 作者保留权利

---

andy stone，工程师

张云龙、徐毛毛、Jianlong Liu 赞同

有一本书好像叫《the art of computer programming》？当年我要做什么算法时通常会去里面找。举几个我记得的:
一、二分法
二、图像快速缩放。有一次我做一个程序，一万张8096x8096的JPEG图片放一起，要求快速放大，缩小，拼接（拼接是另一个算法），全靠从这本书里学的算法。
其他的，想到再说

发布于 2015-05-28　1 条评论 • 作者保留权利

匿名用户

马琦明、izual cc、Argentum 赞同

zkw线段树。空间、时间、代码简洁度都到了一个简直了的地步。

（更）spfa！刷了几道单源最短路的题，全都用了spfa，就是因为快，容易写！bfs版速度快，dfs版判断负环快！简直越刷越精神（๑•̀ㅂ•́）✧

继续刷题...看到了惊艳的就更～

编辑于 2015-06-03　添加评论 • 作者保留权利

邱嘉伟，目前是游戏开发者一枚，热爱互联网文化与…

应江、闲圣、李涛 等人赞同

leetcode 一道题目的最高票答案，看到的时候，真是不明觉厉：
题目链接： Single Number II
答案链接： challenge me , thx

> Given an array of integers, every element appears *three* times except for one. Find that single one.

```
public int singleNumber(int[] A) {
    int ones = 0, twos = 0;
```

```
    for(int i = 0; i < A.length; i++){
        ones = (ones ^ A[i]) & ~twos;
        twos = (twos ^ A[i]) & ~ones;
    }
    return ones;
}
```

发布于 2015-07-26　4 条评论 • 作者保留权利

王钊，初级码农

徐小馗、魔礼沙、思凡wanderer 等人赞同

0x5f3759df
不知道有知道这个梗的没有？

发布于 2015-05-13　25 条评论 • 作者保留权利

井号键，心中有许多愿望，能够实现有多棒。

chaowyc、雷力明、三爽 等人赞同

不用临时变量交换两个数的值

```
a = a ^ b;
b = b ^ a;
a = a ^ b;
```

再补一个蓄水池抽样(reservoir sampling):

在超大数据流的情况下，如何随机抽取其中一行。
因为数据量很大，并不想先读取一次数据，获得数据量n, 然后第二次再根据概率选择数据。

而这个算法的核心思路就是：

所以：
读第一行文件时，保留第一行在内存里，这个概率是 1/1.
读第二行，用1/2的概率来确定这一行要不要替换内存里存着的那一行，
读第三行，用1/3的概率来确定这一行要不要替换内存，
读第四行，用1/4的概率来确定这一行要不要替换内存，
。。。。。
一直读到n行，用1/n的概率来确定这一行要不要替换内存里的那一行。
这个时候，内存里留下的那一行就是随机选取出来的那一行。

不信？你看！

.....

p.s. 这两个算法都是编程珠玑(Programming pearls)里的例子。

另外还有logn 时间复杂度求斐波那契数列.

编辑于 2015-05-18　42 条评论 • 作者保留权利

张狗狗，The devil is in the details

囧玩、顾夕朝、我知道 等人赞同

我是知识的搬运工：

软件正在统治世界。而软件的核心则是算法。算法千千万万，又有哪些算法属于“皇冠上的珍珠”呢？
什么是算法？

> 通俗而言，算法是一个定义明确的计算过程，可以一些值或一组值作为输入并产生一些值或一组值作为输出。因此算法就是将输入转为输出的一系列计算步骤。

—Thomas H. Cormen，Chales E. Leiserson，算法入门第三版
简而言之，算法就是可完成特定任务的一系列步骤，它应该具备三大特征：

> 1、有限
> 2、指令明确
> 3、有效

以下是Marcos Otero推荐的十大算法：
1、归并排序、快速排序及堆积排序

最好的排序算法跟需求密切相关，很难评判。但是从使用上说，这三种的使用频率更高。
归并排序　由冯•诺依曼于1945年发明。这是一种基于比较的排序算法，采用分而治之的办法解决问题，其阶是O(n^2)。
快速排序　可采用原地分割方法，也可采用分而治之算法。这不是一种稳定的排序算法，但对于基于RAM（内存）的数组排序来说非常有效。
堆排序　采用优先级队列来减少数据中的搜索时间。该算法也是原地算法，并非稳定排序。
这些排序算法相对于以前的冒泡排序算法等有了巨大改进，实际上我们今天的数据挖掘、人工智能、链接分析及包括web在内的大多数计算工具都要感谢它们。
2、傅里叶变换与快速傅里叶变换

我们的整个数字世界都使用这两个简单但非常强大的算法　，其作用是将信号从时域转为频域或者反之。实际上，你看得到这篇文章得感谢这些算法。
互联网、你的WiFi、智能手机、电话、计算机、路由器、卫星，几乎所有内置有计算机的东西都会以各种方式使用这两算法。如果不研究这些算法，你就拿不到电子、计算或通信方面的学位。
3、迪杰斯特拉（Dijkstra）算法

Dijkstra 是一种图谱搜索算法。许多问题都可以建模为图谱，然后利用Dijkstra寻找两个节点之间的最短路径。如果没有Dijkstra算法，互联网的运营效率必将大大降低。虽然今天我们已经有了更好的寻找最短路径的解决方案，但出于稳定性的要求，Dijkstra算法仍然被很多系统使用。

4、RSA算法

如果没有密码术和网络安全，互联网就不会像今天一样重要，因为电子商务和电子交易需要这些技术来确保交易安全。而RSA算法 是最重要的密码学算法之一。该算法由同名公司的创始人（Ron Rivest、Adi Shamir和Leonard Adleman）开发，它让密码学普及到了千家万户并奠定了密码术的应用基础。RSA要解决的问题既简单又复杂：如何在独立平台与最终用户之间共享公钥。其解决方案是加密。RSA加密的基础是一个十分简单的数论事实：将两个大素数相乘十分容易，但是想要对其乘积进行因式分解却极其困难，因此可以将乘积公开作为加密密钥。但在分布式计算和量子计算机理论日趋成熟的今天，RSA加密安全性受到了挑战。

5、安全哈希算法（SHA）

这个实际上并不算是算法，而是由美国国家标准技术研究所开发的一系列密码杂凑函数。但是这系列函数是全世界运作的基石。应用商店，电子邮件、反病毒、浏览器等在使用SHA 系列函数，SHA函数可用来确定下载的东西是否自己想要的东西，还是说遭遇了中间人攻击或钓鱼攻击。

6、整数因子分解

这是一个在计算领域使用频繁的数学算法。如果没有这一算法，密码术就会变得不安全得多。整数因子分解是用来将一个合数分解成一系列素因子的一系列步骤。整数因子分解 可被视为是FNP问题（FNP是难以解决的典型NP问题的扩展）。

许多密码协议均基于难以分解的大型合数或相关问题。比方说前面提到的RSA问题。如果有算法能够有效分解任意数字，那么就会使得基于RSA的公钥密码系统陷入不安全的境地。

而量子计算的诞生则令此问题的解决变得容易，从而也打开了一个全新的领域，可利用量子世界的属性来令系统更加安全。

7、链接分析

在互联网时代，不同实体间关系的分析至关重要。从搜索引擎和社交网络到营销分析工具，每个人都想找出互联网的真正结构。

链接分析 无疑是公众对算法的最大困惑与迷思之一。其问题在于进行链接分析有不同的方式，而增加一些特征就会令每一算法略有不同（从而使得算法受到专利保护），但基本上这些算法都是类似的。

链接分析算法首先由Gabriel Pinski和Francis Narin在1976年发明。其背后的思路很简单，即把图谱以矩阵的形式表示，从而转为特征值问题，而特征值有助于了解图谱结构及每个节点的相对重要性。

Google的PageRank，Facebook展示新闻源，Google+，Facebook朋友推荐，LinkedIn工作及联系人推荐，Netflix与Hulu的电影推荐，YouTube视频推荐等均使用了链接分析算法。虽然每个都有不同的目标和参数，但其背后的数学是一样的。

尽管Google似乎是利用此类算法的第一家公司，但是实际上百度创始人李彦宏在Google诞生2两年前做的搜索引擎“RankDex”已经利用这种思路来进行搜索排名了。

8、比例积分微分算法

如果你用过飞机、汽车、微型服务或手机网络，如果你在工厂呆过或者见过机器人，那么你已经见识过这一PID 算法的作用了。

该算法利用了控制回路机制来让期望输出信号与实际输出信号之间的错误降到最小。只要需要信号处理或需要电子系统来控制自动化的机械、水力或热力系统就要用到它。

因此可以说如果没有这一算法，人类的现代文明将不复存在。

9、数据压缩算法

数据压缩算法无疑是非常重要的，因为几乎在所有的结构中都要用到。除了最明显的压缩文档以外，网页下载时也会压缩，视频游戏、视频、音乐、数据存储、云计算、数据库等等也都要使用压缩算法。可以说几乎所有应用都要使用压缩算法。压缩算法令系统更有效成本更低，但是要想确定哪一个最重要却很困难，因为应用不同，使用的压缩算法从zip到mp3、JPEG或MPEG-2各异。

10、随机数生成算法

很多应用都需要随机数。像interlink connection，密码系统、视频游戏、人工智能、优化、问题的初始条件，金融等都需要生成随机数。但实际上目前我们并没有“真正”的随机数生成器，尽管有一些伪随机数生成器也是非常有效的。

当然，十大算法也可能给有凑数之嫌，审视的角度不同对算法的重要性看法也会很不一样，如果你认为这一榜单有错漏的地方，不妨在评论中贡献你的意见。

转自《36氪》 36kr.com/p/212499.html? ...

编辑于 2015-05-14 2 条评论 • 作者保留权利

李盼，engineer

Sirius Caffrey、Shibo Chan 赞同

duff device.

发布于 2015-06-02 添加评论 • 作者保留权利

豆包兔，十八线画手。微信公众号：rabbit-zoo

Sirius Caffrey、赵嫣瑭 赞同

发布于 2015-08-02 1 条评论 • 作者保留权利

马超，PhD student in Computer Science

zivotp、铅笔芯、冬芳 等人赞同

第一次看到遗传算法的时候。虽然现在用的很少了。

发布于 2015-05-12 10 条评论 • 作者保留权利

程穆轩，早知道 是这样，像梦一场，我才不会把爱…

刘梓辰、马路、Liu Steve 等人赞同

看了你的描述，我可以给出我觉得很完美的答案，你们看看是不是的……
牛逼爆了，神步骤……

编辑于 2015-05-13　14 条评论 • 作者保留权利

Claire的钱包，只会逛知乎。

```
template<class T,class U>
struct Node
{
typedef T head;
typedef U next;
};
```

发布于 2015-08-22　添加评论 • 作者保留权利

司空敏慧，算法改变世界
鲍超超、郑志恒、李浩源 等人赞同
第一次看到KMP时

发布于 2015-05-13　8 条评论 • 作者保留权利

李刃，我不想杀人了
unique、铅笔芯、SHUNN 等人赞同

```
int gcd(int x,int y) { return y?gcd(y,x%y):x; }
```

辗转相除法

编辑于 2015-05-19　10 条评论 • 作者保留权利

NAY EUX

杨过 赞同

我一直没发因为我以为早就有人发过了

居然没有？ 那么这个

```
eval(z='p="<"+"pre>"/* ,.oq#+      ,._, */;for(y in n="zw24l6k\
4e3t4jnt4qj24xh2 x/* =<,m#F^    A W###q. */42kty24wrt413n243n\
9h243pdxt41csb yz/* #K       q##H######Am */43iyb6k43pk7243nm\
r24".split(4)){/* dP      cpq#q##########b, */for(a in t=pars\
eInt(n[y],36)+/*         p##@###YG=[#######y */(e=x=r=[]))for\
(r=!r,i=0;t[a/*         d#qg `*PWo##q#######D */]>i;i+=.05)wi\
th(Math)x-= /*        aem1k.com Q###KWR#### W[ */.05,0>cos(o=\
new Date/1e3/*      .Q#########Md#.###OP  A@ , */+x/PI)&&(e[~\
~(32*sin(o)*/* ,     (W######Xx######.P^     T % */sin(.5+y/7))\
+60] =-~ r);/* #y     ``^TqW####P###BP           */for(x=0;122>\
x;)p+="   *#"/* b.        OQ####x#K           */[e[x++]+e[x++\
]]||(S=("eval"/* l         `X#####D  ,       */+"(z='"+z.spl\
it(B = "\\\\").join(B+Q/*          G####B" #       */join(B+B).split\
(Q="\'").join(B+Q/*          VQBP`        */)+Q+")//m1k")[x/2\
+61*y-1]).fontcolor/*         TP         */(/\\w/.test(S)&&"#\
03B");document.body.innerHTML=p+=B+"\\n"}setTimeout(z)')//
```

会转的哟~

编辑于 2015-08-22 4 条评论 • 作者保留权利

姜小豆，想成为程序员的男人

Sirius Caffrey、徐毛毛、我姓封疯子的疯 等人赞同

int(number+0.5)

四舍五入

发布于 2015-05-13 9 条评论 • 作者保留权利

紫电，计算机/ACM/现代史

陈淇、郭雨、归去来兮迟 等人赞同

遗传算法。

仅仅只是通过简单的基因交换和变异，就筛选出最优解。

第一次看完之后，感觉生物千百万年的进化发展尽在于此，由衷的感叹这个世界的美妙。看世界的方式都不一样了，三观也再次被刷新——

1、基本制度的重要性。什么样的函数，筛选出什么样的结果。

2、近亲结婚也是有优点的←_←

编辑于 2015-08-22 添加评论 • 作者保留权利

Json

Tomoya Okazaki、夏清、forcey 等人赞同

```
float Q_rsqrt( float number )
{
 long i;
 float x2, y;
 const float threehalfs = 1.5F;

 x2 = number * 0.5F;
 y  = number;
 i  = * ( long * ) &y;                       // evil floating point bit level hacking
 i  = 0x5f3759df - ( i >> 1 );               // what the fuck?
 y  = * ( float * ) &i;
 y  = y * ( threehalfs - ( x2 * y * y ) );   // 1st iteration
//      y  = y * ( threehalfs - ( x2 * y * y ) );   // 2nd iteration, this can be removed

 return y;
}

//关键在于  what the fuck 那句代码。
```

编辑于 2015-05-13 1 条评论 • 作者保留权利

吴文化

前段时间看了用一个魔数，利用牛顿迭代求根号，当时就震惊了

发布于 2015-07-27 添加评论 • 作者保留权利

Break，码农

其实汇编与机器代码都是人类高等的智慧，物希为奇，目前能把一张图的机器码写出来的好像没见过

发布于 2015-08-01 添加评论 • 作者保留权利

飒飒

邵烟球、李思睿 赞同

卷积神经网络！

发布于 2015-06-07 添加评论 • 作者保留权利

永怀，长头发的程序员

蒋思淼、小栗子和大钻风、瓦尔基里Valkyrja 等人赞同

Raid5磁盘阵列的恢复算法，N块磁盘组成Raid5阵列，其中一块会被拿来存储其他盘的校验位，也就是其他N-1块盘同一位置的数据异或得到的结果，当任意一块盘的数据损坏时，都能根据其他盘的数据结合校验位恢复出这块盘的内容来，校验盘自己坏掉了也能使用其他盘的数据重新计算出校验盘的内容。当然，同时坏任意两块的话，所有盘的数据就都完蛋了，但是正常工作状况下两块硬盘同时坏掉的概率显然比一块要小太多。
这个算法让我惊叹的地方在于，用异或这样一个无比简单的运算，实现了最低成本的数据备份策略（一块硬盘保护多块硬盘的数据）

编辑于 2015-05-19 添加评论 • 作者保留权利

魏洲，掏粪男孩，CAFEBABE

卢小树苗、乙醚 赞同

东尼·霍尔 的快速排序

发布于 2015-05-14 1 条评论 • 作者保留权利

王逸翀TeX，写代码的 拍照片的

请题主去 kernel.org 上找任意一个版本的Linux内核然后随意打开里面一段代码即可

发布于 2015-08-28 添加评论 • 作者保留权利

孟奇，做音乐,做乐器,合成器与Max的教师.

Fletin、Jimmy Zhang 赞同

m.youtube.com/watch? ...
由4-mat创作，23字节，9分19秒还带结构的demo。

另外话说demoscene很多精品。推荐 pouet.net

发布于 2015-07-31 添加评论 • 作者保留权利

zy Wen，误入艺术领域的伪geek

张白虹、kiter、袁宇 等人赞同

刚好想到一段，不敢称璀璨无比，仅仅觉得有趣，值得玩味。

一段可以囊括世间万物的代码。

```
void setup(){
size(500,500);
}
void draw(){
for(int i=0;i<width;i++){
for(int j=0;j<height;j++){
stroke(frameCount/pow(255,i+j*width)%255,frameCount/pow(255,i+j*width+1)%255,frameCount/pow(255,i+j*width+2)%255);
point(i,j);
}
}
```

```
}
```

语言用的是 Processing 。期间程序只做一件事，那就是穷举。

```

```

忽略机器本身的性能限制，假设 frameCount 可以无限大（frameCount代表当前帧数）。
你只需安安静静地盯着屏幕，就可以看到所有像素的所有组合可能。

这意味着你可以在上面看到所有艺术大师的作品，以及人类历史上所有光辉的瞬间。同时还有这个回答下的任何图片，只要你有足够的时间。

这个时间得多长？我们可以简单地计算一下。

计算机里的每个像素都是由 256 级的 RGB 组成的。因此可以表示 256^3 （1600万）种颜色。假如图形的分辨率为 1000 × 1000，所有的像素可能的色值相互组合，将会产生 256 的 3000000 次方张不同图片。

我们如果将图片排在一个长廊上，人以一秒一张的速度去浏览。由于一年有 31536000 秒，因此要走完这条长廊，就需要 10^{2212} 年。

```

```

这个数字已经大得很难用人类的常用单位去表示了。硬是要用，那就是 10亿亿亿亿亿亿.....年（90万个亿）。要清楚，宇宙的年龄也仅仅是 140 亿年（1.4×10^1 年）。。

这也意味着，即使你从宇宙大爆炸看到现在，也无法将这个程序看完。

但如果把图片像素的精度降低呢？用 100 × 100 的分辨率并且只用黑白二值去表示图形？此时总数就会缩减到 2^{10000} ，也就约等于 10^{3011} 。

看似缩小很多了。如果同时动用全人类的力量，将这个任务分配给70亿人。每人还是要不眠不休地看上 3.17×10^{3002} 年，才能看完。

即使化到最简，结果仍是大得恐怖。但如果能看完，我手上说不准会有一张 100 × 100 的HelloKitty头像，他手上或许能有一张爱因斯坦吐舌头的照片。

可以用这么简洁的形式去展现万物，用近乎无限的时间去换取无限的可能，我觉得这就是这段代码的魅力所在。

————————————————————————————

感兴趣的可以看一个 5 × 5 的精简加速版感受图形变化的过程。（上面的算法在开始时，跑起来的大部分时间是黑漆漆一片的）

```
int num,w,frame,level;

void setup(){
size(400, 400);
num = 5;
w = width/num;
level = 2; //色值精度
}

void draw(){
for(int i = 0; i < num; i++){
for(int j = 0; j < num; j++){
fill((int(frame/pow(level,i + j * num)) % level)* (255 / (level - 1)));
rect(w * i, w * j, w, w);
}
}
// frame++; 匀速播放
frame = int(pow(frameCount,2)); //加速播放
}
```

可以尽情地修改各种参数：）

End

编辑于 2015-08-26 添加评论 • 作者保留权利

Jaime， Just just just a lazy man

深度学习…

发布于 2015-05-16 添加评论 • 作者保留权利

朱卫斌，志存高远 脚踏实地

Zagfai Kwong、张尼玛 赞同

while(1) { printf("啪") }

发布于 2015-05-30 2 条评论 • 作者保留权利

taloyum casa，反革命攻城狮

```
while(x-->0)
{
print x;
}
```

发布于 2015-08-04 添加评论 • 作者保留权利

肖剑锋，为了让代码更简洁，砍了需求方

Jing Guo、朱Steven、francium bobo 赞同

没有人提到 fork炸弹吗？

zh.m.wikipedia.org/wiki ...

发布于 2015-05-17 添加评论 • 作者保留权利

Vct Marine，前军校学员，努力打入MSRA，AR游戏

矿里掘金、林飞、李强 赞同

第一次见支持向量机，以及佛洛伊德算法

发布于 2015-05-14 1 条评论 • 作者保留权利

时冰蓝

Y combinator.

完美递归隐藏于完美递归.欺骗解释器,欺骗世界.

发布于 2015-05-31 添加评论 • 作者保留权利

周源，无处话凄凉

邵烟球、清水古木、侯良 等人赞同

#define TRUE FALSE

还有define public private

编辑于 2015-05-14 5 条评论 • 作者保留权利

Wei Ma，Transportation PhD, 求开闭乃团二次元狗…

徐素素、畅所欲言、赵小翔 等人赞同

我觉得我需要说的是一种更原始的“代码”， Turing Machine, 图灵机完成了对computation的建模，也就是说，对于现在世界上的所有代码，你都可以写（做）一个图灵机来模拟它。而图灵机所需要的仅仅是一个无限长的纸条和笔还有想橡皮罢了。

图灵机真正告诉了人们什么叫做Computation, 也界定了什么代码能够实现什么代码不能够实现。这种近乎数学乃至哲学意义上的“代码”可谓是人类智慧的完美结晶。

PS。图灵机在computational complexity领域也有极重要的地位，图灵真可谓是天才。。只可惜他是同性恋，然而当时英国政府是不允许同性恋的，因此强迫他服用了很多精神类的药物，最后他自杀了。。真希望他能多活几年。。。。

PSS: 在图灵机之后，有很多对computation更加恐怖的抽象，例如，如果你只有3个counter （计数器，可以从1数到无穷）， 你就已经可以模拟任何代码。。。。也就是说不论多么复杂的代码可以被抽象到三个counter数字的增减。。是不是匪夷所思。。。。这也是拜图灵机所提出的computation是localized这个特性所赐。。

PSSS:直到量子计算提出，图灵机的统治地位才收到了一点动摇。。。。

发布于 2015-05-13 4 条评论 • 作者保留权利

DrEdgar，.

陈晟祺、张瑜、古慎龙 等人赞同

code.activestate.com/re ...

利用 Python 的复数计算支持，三行解一元一次方程。

发布于 2015-05-11　1 条评论 • 作者保留权利

Maples7，EE2CS

温和林、4760947779、史海波 等人赞同

1.

C++中对后置++的重载：

```
Clock Clock::operator ++ (int) {
    //注意形参表中的整型参数
    Clock old = *this;
    ++(*this);  //调用前置“++”运算符
    return old;
}
```

Clock类是自己声明的。

看到重载源码瞬间就可以理解前置++和后置++的计算过程。
操作原对象，返回操作前对该对象的副本。所以在使用时用的是原值（实际上是副本），而使用后原对象本身已经自加了。

2.

计算二进制中“1”的个数：

```
#define POW(c) (1<<(c))
#define MASK(c) (((unsigned long)-1) / (POW(POW(c)) + 1))
#define ROUND(n, c) (((n) & MASK(c)) + ((n) >> POW(c) & MASK(c)))

int bit_count(unsigned int n)
{
    n = ROUND(n, 0);
    n = ROUND(n, 1);
    n = ROUND(n, 2);
    n = ROUND(n, 3);
    n = ROUND(n, 4);
    return n;
}
```

编辑于 2015-05-30　2 条评论 • 作者保留权利

更多

22 个回答被折叠 （为什么？）

ywang，**穷酸学生**　修改话题经验

写回答…

匿名 | 未经许可，禁止转载

发布回答

关注问题

9227 人关注该问题

相关问题

python 一个类的函数如何调用另一个类中函数的返回值？ 1 个回答

问题状态

最近活动于 01:16 • 查看问题日志
被浏览 674468 次，相关话题关注者 258214 人

© 2015 知乎