程序设计实践
The Practice of Programming

风格(Style)

概述

代码风格很多因程序员的好恶而异。另一方面，技术的发展也使得一些代码风格要求在发生变化。

对于有争议的“风格”，我们应该：

- 遵从你所在组织的规范。
- 与同一个文件中的其它代码风格保持一致。
- 与同一个项目里其它文件的风格尽量保持一致。

本章涉及的风格，应该是一些无争议的风格，例如：命名风格，表达式和语句的风格，一致性和习惯用法，函数宏定义，避免神秘的数，注释风格等。

雷军当程序员时写的代码（汇编语言）

```
;
; RI.ASM Revision 2.12 [ July 12, 1994 ]
Revision equ 'V2.12 '
;
;
;
;  RAMinit Release 2.0
;  Copyright (c) 1989-1994 by Yellow Rose Software Co.
;  Written by Mr. Leijun
;
;  Function:
;  Press HotKey to remove all TSR program after this program
;
;

; ..............................................................
; Removed Softwares by RI:
; SPDOS v6.0F, WPS v3.0F
; Game Busters III, IV
; NETX ( Novell 3.11 )
; PC-CACHE
; Norton Cache
; Microsoft SmartDrv
; SideKick 1.56A
; MOUSE Driver
; Crazy (Monochrome simulate CGA program)
; RAMBIOS v2.0
; 386MAX Version 6.01
; ..............................................................
; No cancel softwares:
; Windows 3.1 MSD
```



```
1084  Main0:
1085  cld
1086  Print instMsg
1087  call IsWinDos
1088  or ax, ax
1089  jz @@1
1090  Print WinErr
1091  mov ax, 4c00h
1092  int 21h
1093  @@1:
1094  call HotKeyValid
1095  mov cs:Status, 0
1096  call EMS_test
1097  call CmpDosVer
1098  call CmpSideKick
1099  call GetMachineID
1100  call ModifyHotKeyPrompt
1101
1102  mov ax, 0c0d7h
1103  int 2fh
1104  mov es, ax
1105  cmp word ptr es:[101h], 'IE' ; 'LEI'
1106  jnz @@0
1107  mov cs:Self, ax
1108  @@0:
1109  call CmdLine
1110  call PrintHotKeyPrompt
```

写代码类似于写文章,要清晰（clear）而简洁（concise）。

What does writing clearly and concisely mean?

Writing clearly and concisely means choosing your words deliberately and precisely, constructing your sentences carefully to eliminate deadwood, and using grammar properly. By writing clearly and concisely, you will get straight to your point in a way your audience can easily comprehend.

写得清楚、简洁意味着要有意识、准确地选择单词，仔细地构造句子以消除枯燥，并正确地使用语法。通过清晰、简洁的写作，你将以一种你的听众可以轻松理解的方式直奔主题。

Why should I write clearly and concisely?

To succeed in your communication, you need to keep your audience's attention, and your audience needs to read through documents effortlessly and with understanding. If your writing is difficult to follow, your readers may lose interest (and patience).

要想在沟通中取得成功，你需要保持听众的注意力，而听众需要轻松、理解地阅读文档。如果你的文章很难理解，读者可能会失去兴趣（和耐心）。

来源：https://stanford.edu/class/ee267/WIM/writing_style_guide.pdf

命名的风格

```
#define ONE 1
#define TEN 10
#define TWENTY 20
```

```
#define INPUT_MODE 1
#define INPUT_BUFSIZE 10
#define OUTOUT_BUFSIZE 20
```

名字要准确表达用途

试想，如果一个有TWENTY个元素的数组想改得包含更多元素，简单地把数值20改大虽然能让程序运行，但是却易于误导代码的阅读者。

```
for (elementIndex = 0; elementIndex < numberOfElements;
       elementIndex++)
   elementArray[elementIndex] = elementIndex;
```

```
for (i = 0; i < nitems; i++)
   elem[i] = i;
```

名字不是越长越好。

全局变量可以用具有描述意义的长名字；
局部变量用短名字会显得程序更简洁。

```
Class UserQueue{
    int numOfItemsInQ, frontOfQueue, queueCapacity;
    public int numOfUsersInQueue() {...}
```

```
Class UserQueue{
    int nUsers, front, capacity;
    public int getNumOfUsers() {...}
}
```

类的成员的名字，不要再啰嗦重复类名。

```
now = date.getTime()
```

```
putchar('\n')
```

用动词或者动词+名词定义函数名。

```
#define checkoctal(c) ((c) >= '0' && (c) <= '7')
if (checkoctal(x))
    ...
```

```
#define isoctal(c) ((c) >= '0' && (c) <= '7')
if (isoctal(x))
    ...
```

返回布尔值的函数名应该是一个命题，通常使用is作为函数名中的动词。

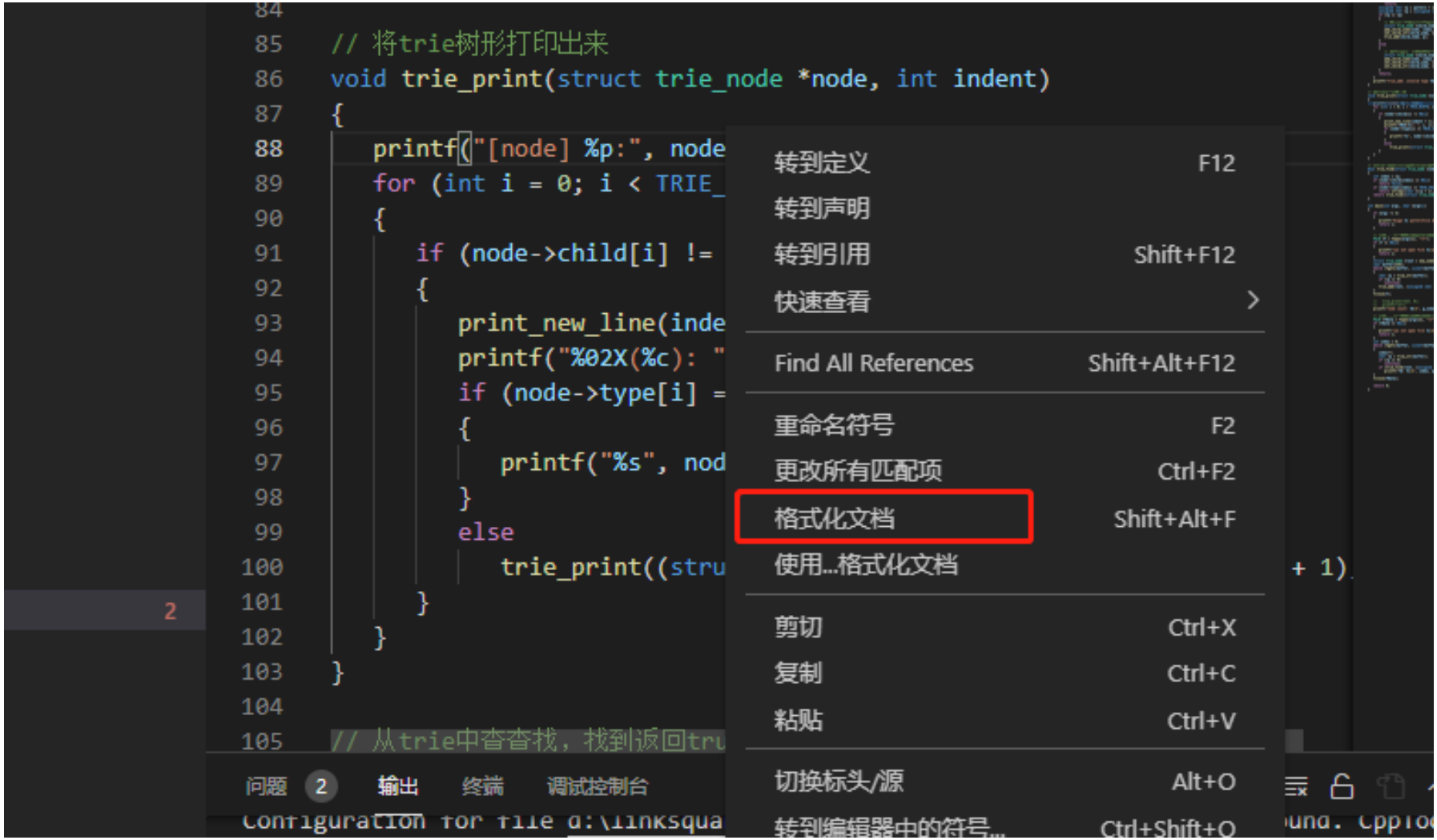
```
bool not_eof = true
if((ch = getchar()) == EOF)
    not_eof = false;
```

```
bool is_eof = false;
if((ch = getchar()) == EOF)
    is_eof = true;
```

名字尽可能简单明了，不要用“否定之否定”。

表达式和语句

- 使用缩进来显示程序结构
- 大多数集成开发环境(IDE)都提供代码格式化功能
- 阅读代码前，可以先做代码格式化



```
if(!(block_id < actblks) || !(block_id >= unblocks))
    ...
```

```
if((block_id >= actblks) || (block_id < unblocks))
    ...
```

使用表达式的自然形式，即常见的易于理解的形式。

```
#define MASK 0x0F
#define BITS 0x01
if(x & MASK == BITS)
    ...
```

```
#define MASK 0x0F
#define BITS 0x01
if((x & MASK) == BITS)
    ...
```

利用括号排除歧义。

即使程序执行上不需要括号，例如

```
leap-year = y % 4 == 0 && y % 100 != 0 || y % 400 == 0;
```

也建议加上括号以便提高代码可读性

```
*x += (*xp = (2 * k < (n - m) ? c[k + 1] : d[k--]));
```

```
if (2 * k < (n - m))
    *xp = c[k + 1];
else
    *xp = d[k--];
*x += *xp
```

分解复杂的表达式。

练习

```
if(!(c == 'y' || c == 'Y'))
    return
```

```
if((c != 'y') && (c != 'Y'))
    return
```

```
length = (length < BUFSIZE) ? length : BUFSIZE
```

```
if(length > BUFSIZE)
    length = BUFSIZE
```

```
flag = flag ? false : true
```

```
flag = !flag
```

```
quote = (*line == '"') ? true : false
```

```
quote = *line == '"'
```

```
quote = (*line == '"')
```

一致性和习惯用法

```
if(month == FEB){
    if(year%4 == 0)
        if(day > 29)
            legal = FALSE;
    else
        if(day > 28)
            legal = FALSE;
}
```

```
if(month == FEB){
    if(year%4 == 0){
        if(day > 29)
            legal = FALSE;
    }else{
        if(day > 28)
            legal = FALSE;
    }
}
```

使用一致的缩进和花括号风格（例如两行以上一定加花括号）

```
i = 0;
while(i <= n-1)
    array[i++] = 1.0;
```

```
for(i = 0; i < n; )
    array[i++] = 1.0;
```

```
for(i = n; --i >= 0; )
    array[i] = 1.0;
```

```
for(i = 0; i < n; i++)
    array[i] = 1.0;
```

使用大多数人最习惯用法。
（for循环设计的目的，就是让循环的起始状态，结束条件和步长一目了然）

```
if(argc == 3)
   if(fin = fopen(argv[1], "r")) != NULL)
      if(fout = fopen(argv[2]. "w")) != NULL){
         while((c = getc(fin)) != EOF)
            putc(c, fout);
         fclose(fin);
         fclose(fout);
      }else
         printf("Can't open output file %s\n",argv[2]);
   else
      printf("Can't open input file %s\n", argv[1]);
else
   printf("Usage: cp inputfile outputfile\n");
```

```
if(argc != 3)
   printf("Usage: cp inputfile oputputfile\n");
else if(fin = fopen(argv[1], "r")) == NULL)
   printf("Can't open input file %s\n", argv[1]);
else if(fout = fopen(argv[2]. "w")) != NULL){
   printf("Can't open output file %s\n", argv[2]);
   fclose(fin);
}else{
   while((c = getc(fin)) != EOF)
      putc(c, fout);
   fclose(fin);
   fclose(fout);
}
```

用else-if 处理多路选择，
一个判断应尽可能接近它所对应的动作

练习

```
for(k = 0; k++ < 5; x += dx)
    sscanf("%lf", &dx);
```

```
for(k = 0; k < 5; k++)
{
    sscanf("%lf", &dx);
    x += dx;
}
```

练习

```
int count = 0;
while(count < total){
    if(this.getName(count) == nameTable.userName()) {
        return true;
    }
    count++;
}
```

```
for(int count = 0; count < total; count++) {
    if(this.getName(count) == nameTable.userName())
        return true;
}
```

函数宏

```
#define isupper(c) ((c) >= 'A' && (c) <= 'Z')
while((c = getchar()) >= 'A' && (c = getchar()) <= 'Z')
    ...
```

```
/*** ctype.h中定义的isupper更好，参数只出现一次 ***/
# define __isctype(c, type) \
  ((*__ctype_b_loc())[(int)(c)] & (unsigned short)type)
# define isupper(c)     __isctype((c), _ISupper)
```

```
/*** 避免isupper的参数中写很多东西 ***/
while((c = getchar()) != EOF && isupper(c))
    ...
```

函数宏的缺点，直接替换，导致用到两次的参数会被替换两次，执行两次，可能直接导致错误。

isupper中的(c)参数出现了两次，容易导致问题，解决办法有两个：

1. isupper定义的好一点，(c)只出现一次，即使第一个例子那样调用也不会有问题；
2. 调用isupper的时候注意，不要用复杂的参数代入(c)。

```
#define ROUND_TO_INT(x) ((int)((x)+(((x)>0)?0.5:-0.5)))

size = ((int)((sqrt(x)+(((sqrt(x)>0)?0.5:-0.5)))
```

```
int round_to_int(float x)
{
   return (int)(x + ((x > 0) ? 0.5 : -0.5));
}
size = round_to_int(sqrt(x));
```

函数宏的缺点，直接替换，导致用到两次的参数会被替换两次，执行两次，即便功能正确，也可能导致性能问题。

```
#define square(x) (x) * (x)
y = 1 / (x) * (x)
```

```
#define square(x) ((x) * (x))
y = 1 / ((x) * (x))
```

如果一定要用函数宏，不要吝啬括号

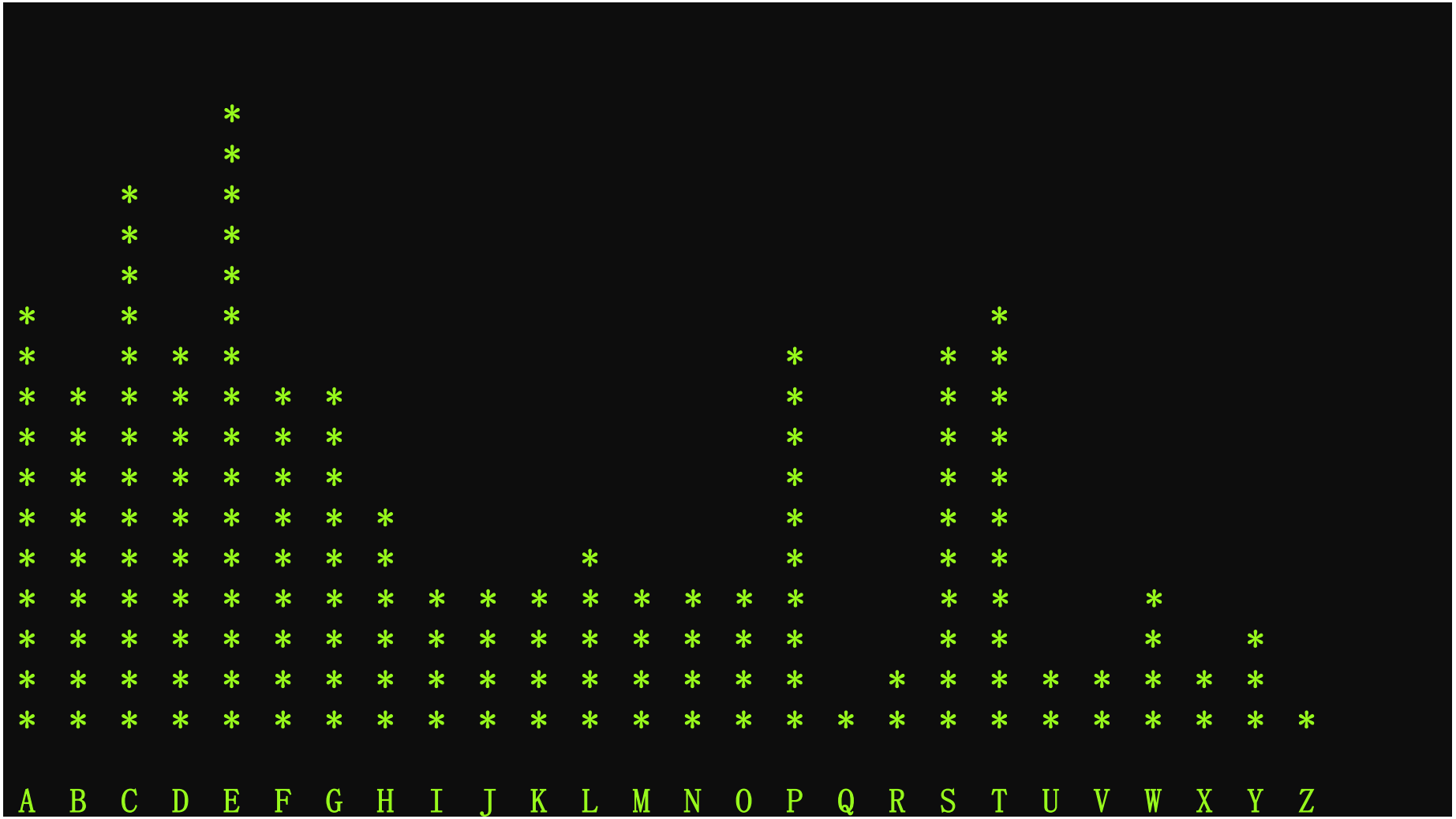
函数宏小结

- C++可以尽可能使用inline函数
- Java则根本不支持函数宏
- C语言尽量不要使用函数宏
- 如果一定要使用，那么注意：
  - 函数体和参数都加括号
  - 尽量不要代入复杂的参数

神秘的数

```
fac = lim/20;      /* set scale factor */
if(fac < 1)
    fac = 1;
/* generate histogram */
for(i = 0, col = 0; i < 27; i++,j++){
    col + = 3;
    k = 21 - (let[i] / fac);
    star = (let[i] == 0)?' ':'*';
    for(j = k; j < 22; j++)
        draw(j, col, star);
}
draw(23, 2, ' '); /* label x axis */
for(i = 'A'; i <= 'Z'; i++)
        printf("%c  ", i);
```

这么多神秘的数都是什么意思？



这个程序是要实现这样一个直方图的打印

```
enum{
    MINROW     = 1,                    /* top edge */
    MINCOL     = 1,                    /* left edge */
    MAXROW     = 24,                   /* bottom edge(<=) */
    MAXCOL     = 80,                   /* right edge(<=) */
    LABELROW   = 1,                    /* position of labels */
    NLET       = 26,                   /* size of alphabet */
    HEIGHT     = MAXROW - 4,/* height of bars */
    WIDTH      = (MAXCOL - 1)/NLET /* width of bars */
};
fac = (lim + HEIGHT-1)/HEIGHT; /* set scale factor */
if(fac < 1)
    fac = 1;
/* generate histogram */
for(i = 0; i < NLET; i++){
    if(let[i] == 0)
        continue;
    for(j = HEIGHT - let[i]/fac; j < HEIGHT; j++)
        draw(j+1+LABELROW, (i+1)*WIDTH, '*');
}
draw(MAXROW-1, MINCOL+1, ' ');    /* label x axis */
for(i = 'A'; i <= 'Z'; i++)
    printf("%c  ", i);
```

```
char buf[1024];
fgets(buf, 1024, stdin)
```

```
#define BUF_SIZE 1024
char buf[BUF_SIZE];
fgets(buf, BUF_SIZE, stdin);
```

```
char buf[1024];
fgets(buf, sizeof(buf), stdin);
```

利用语言去计算对象的大小，从而减少代码修改负担，提高代码简洁性。

```
int intArray[1024];
for(int i = 0; i < sizeof(intArray); i++)
    ...
```

```
int intArray[1024];
for(int i = 0; i < sizeof(intArray) / sizeof(int); i++)
    ...
```

```
#define NELEMS(array) (sizeof(array) / sizeof(array[0]))

int intArray[1024];
for(int i = 0; i < NELEMS(intArray); i++)
    ...

double doubleArray[1024];
for(int i = 0; i < NELEMS(doubleArray); i++)
    ...
```

利用语言去计算对象的大小，可以定义一个通用的宏。

练习

```
#define FT_TO_METER     0.3048
#define METER_TO_FT     3.28084
#define MILE_TO_KM      1.609344
#define SQMILE_TO_SQKM 2.589988
```

```
#define FT_TO_METER     0.3048
#define METER_TO_FT     (1 / FT_TO_METER)
#define MILE_TO_KM      1.609344
#define SQMILE_TO_SQKM (MILE_TO_KM * MILE_TO_KM)
```

注释的风格

```
if((country == SING) || (country == BRNI) ||
    (country == POL) || (country == ITALY))
{
    /* If the country is Singapore, Brunei or Poland
     * then the current time is the answer time
     * rather than the off hook time
     * Reset answer time and set day of week
     */
    ...
}
```

注释的目的是解惑，而不是增加疑惑。
注释与代码要始终保持一致。

```
/*Add Amount to Total*/
total = amount + total
```

```
if (c == '(') /* left paren */
    ...
else if (c == ')') /* right paren */
    ...
else if (c == ';') /* semicolon */
    ...
```

不要写没用的注释。

```
...
/* if "result" is 0 a match was found so return true
  Otherwise, "result" is non-zero so return false */
return (!result)
```

```
...
return matchFound;
```

用好的命名代替注释。

用注释自动生成文档

```
/**
 * @function rand
 * 返回一个随机数。
 * @return number - 返回的随机数
 * @examples
 * # 例如要求得0-999之间的随机数，可以这样写:
 * n = rand() % 1000
 */
int execFunc_rand(CUniScript* script, CExpResultList&
    argList, CExpResult& result)
{
    result.setNumber(rand());
    return R_CONTINUE;
}
```

用注释自动生成文档

```
$ uslhelp rand

NAME
    rand

DESCRIPTION
    返回一个随机数。

RETURN VALUES
    number - 返回的随机数

EXAMPLES
    # 例如要求得0-999之间的随机数，可
    n = rand() % 1000
```

rand

DESCRIPTION:

返回一个随机数。

PARAMETERS:

void

RETURN VALUE:

number : 返回的随机数

EXAMPLES:

```
# 例如要求得0-999之间的随机数，可以这样写:
n = rand() % 1000
```

用javadoc生成HTML说明文档举例

SquareNum.java 文件代码:

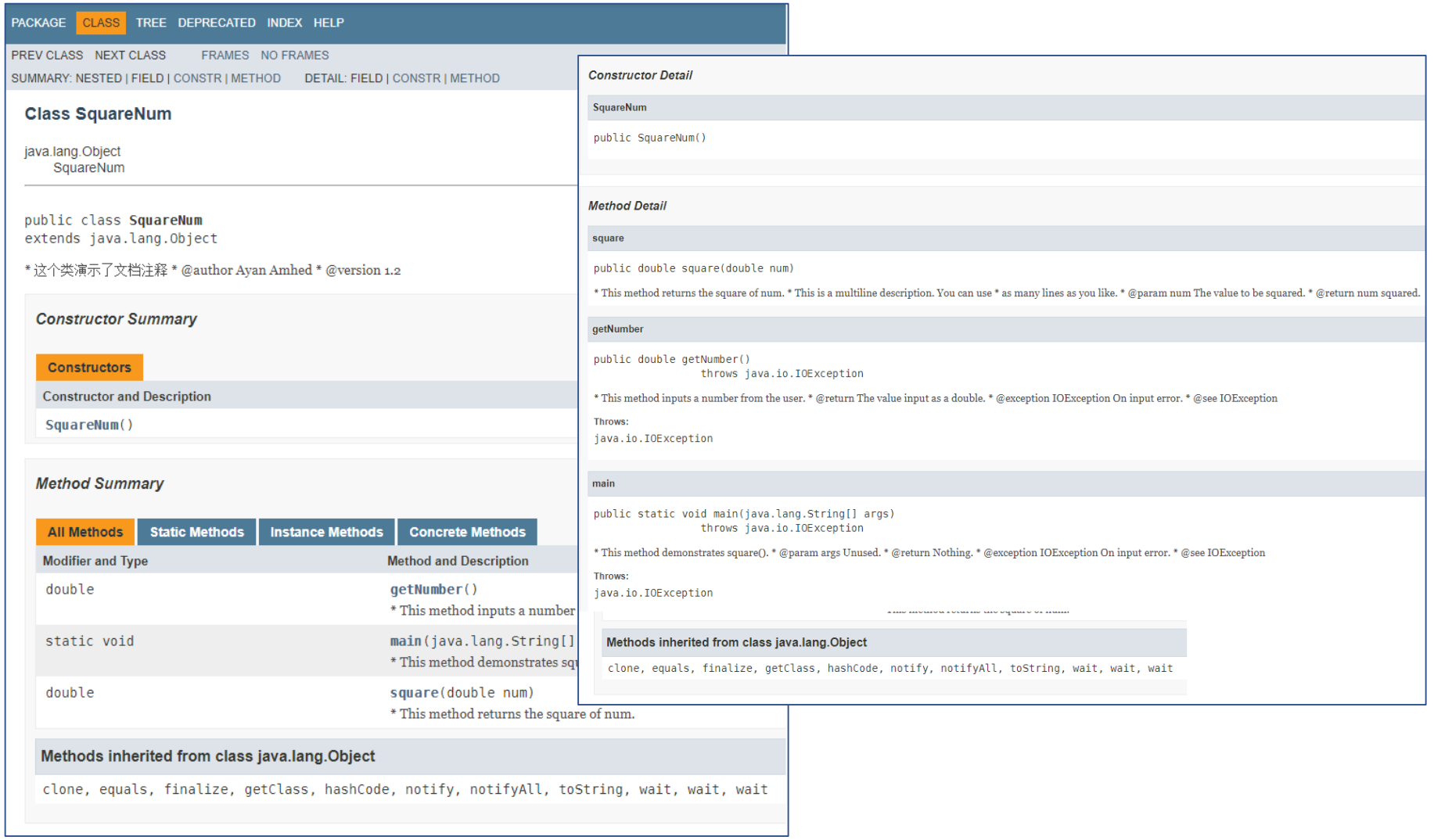
```
import java.io.*;

/**
* 这个类演示了文档注释
* @author Ayan Amhed
* @version 1.2
*/
public class SquareNum {
   /**
   * This method returns the square of num.
   * This is a multiline description. You
can use
   * as many lines as you like.
   * @param num The value to be squared.
   * @return num squared.
   */
   public double square(double num) {
      return num * num;
   }
   /**
   * This method inputs a number from the
user.
   * @return The value input as a double.
   * @exception IOException On input error.
   * @see IOException
   */
public double getNumber() throws IOException
{
      InputStreamReader isr = new
InputStreamReader(System.in);
      BufferedReader inData = new
BufferedReader(isr);
      String str;
      str = inData.readLine();
      return (new
Double(str)).doubleValue();
   }
   /**
   * This method demonstrates square().
   * @param args Unused.
   * @return Nothing.
   * @exception IOException On input error.
   * @see IOException
   */
   public static void main(String args[])
throws IOException
   {
      SquareNum ob = new SquareNum();
      double val;
      System.out.println("Enter value to be
squared: ");
    val = ob.getNumber();
      val = ob.square(val);
      System.out.println("Squared value is "
+ val);
   }
}
```

执行如下的指令： javadoc SquareNum.java

源代码文件

生成后的HTML页面效果



PACKAGE CLASS TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class SquareNum

java.lang.Object
SquareNum

```
public class SquareNum
extends java.lang.Object
```

* 这个类演示了文档注释 * @author Ayan Amhed * @version 1.2

Constructor Summary

Constructors

Constructor and Description
SquareNum()

Method Summary

All Methods | Static Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
double	getNumber() * This method inputs a number
static void	main(java.lang.String[] * This method demonstrates sq
double	square(double num) * This method returns the square of num.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SquareNum

```
public SquareNum()
```

Method Detail

square

```
public double square(double num)
```

* This method returns the square of num. * This is a multiline description. You can use * as many lines as you like. * @param num The value to be squared. * @return num squared.

getNumber

```
public double getNumber()
                throws java.io.IOException
```

* This method inputs a number from the user. * @return The value input as a double. * @exception IOException On input error. * @see IOException

Throws:
java.io.IOException

main

```
public static void main(java.lang.String[] args)
                throws java.io.IOException
```

* This method demonstrates square(). * @param args Unused. * @return Nothing. * @exception IOException On input error. * @see IOException

Throws:
java.io.IOException

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

总结

命名要准确表达用途。
名字不是越长越好。一般来说，全局变量用具有描述意义的长名字，局部变量用短名字会显得程序更简洁。
类里面的成员的名字，不要再重复类名，啰嗦。
用动词或者动词+名词定义函数名。
返回布尔值的函数名应该是一个命题，不应该是一个动作。
命名尽可能简单明了，不要用“否定之否定”。

使用表达式的自然形式。
利用括号排除歧义。
分解复杂的表达式。
不要简单问题复杂化。

使用一致的缩进和加括号风格。
为了一致性，使用习惯用法。
用else-if 处理多路选择，一个判断应尽可能接近它所对应的动作。

函数宏存在缺点，尽量不要使用它。
必须要使用的时候，给宏的体和参数都加上括号。

给神秘的数起个名字。
利用语言去计算对象的大小。

注释是为了解惑，而不是带来疑惑。
注释与代码要始终保持一致。
不要写没用的注释。
用好的命名代替注释。
用注释自动生成文档。

课下自习：

阿里代码规约

https://github.com/alibaba/p3c

PMD - source code analyzer

https://github.com/pmd/pmd

It finds common programming flaws like unused variables, empty catch blocks, unnecessary object creation, and so forth. It supports many languages. It can be extended with custom rules. It uses JavaCC and Antlr to parse source files into abstract syntax trees (AST) and runs rules against them to find violations. Rules can be written in Java or using a XPath query.