生活中的分治策略和计算机语言

分治策略是将问题逐步分解为一些易于解决的子问题，然后再将每个子问题各个击破的过程。

计算机执行特定任务是通过执行预定义的指令集来实现的。这些预定义的指令集就是所谓的计算机程序。

计算机能“理解”的唯一语言就是机器语言机器语言是由一系列二进制的0和1组成的。由于机器语言很难直接使用,所以通常将计算机指令表示成一种特殊的语言,这种特殊的语言被称为程序设计语言。

程序设计语言不是机器语言,机器语言是一种低级语言,而程序设计语言则是一种高级语言。使用高级语言来编写程序比使用低级语言容易得多。程序设计语言又分为面向过程语言，和面向对象语言(new)。

为了让计算机执行由高级语言编写的程序指令,必须把这些指令从高级语言形式转换成计算机能理解的机器语言形式,这种转换是由编译器来完成的。

算法的概念及其描述方法

算法是为解决一个具体问题而采取的确定，有限，有序，可执行的操作步骤。计算机算法仅表示计算机能够执行的算法。

程序设计是一门艺术，主要体现在算法设计和结构设计上，如果说结构设计是程序的肉体，那么算法设计就是程序的灵魂。

计算机科学家沃思曾经提出公式。
数据结构+算法=程序

这个公式仅对面向过程的语言(如C语言)成立,它说明一个程序应由两部分组成:
(1)数据结构(Data Structure)是计算机在储、组织数据的方式,指相互之间存在一种或多种特定关系的数据元素的集合。
(2)算法是对操作或行为(即操作步骤)的描述。算法代表着用系统的方法摘述解,决问题的策略。不同的算法可能用不同的时间、空间或效率来完成同样的任务。计算机进行问题求解的算法大致可分为如下两类:

(1)数值算法,主要用于解决数值求解问题。
(2)非数值算法,主要用于解决需要用逻辑推理才能解决的问题,如人工智能中的许多问题以及搜索、分类等问题都属于这类算法。
那么怎样衡量一个算法的正确性呢?一般的,可用如下基本特性来衡量。
(1)有穷性。算法包含的操作步骤应是有限的,每一步都应在合理的时间内完成,否则算法就失去了它的使用价值。
(2)确定性。算法的每个步骤都应是确定的,不允许有歧义。例如,“如果x20,则输出Yes;如果xs0,则输出No"就是有歧义的,即当 等于0时,既要输出Yes,又要输出,No,这就产生了不确定性。
(3)有效性,算法中的每个步骤都应能有效执行,且能得到确定的结果。例如,对,,个负数开平方或者取对数,就是一个无效的操作。
(4)允许没有输人或者有多个输人。有些算法不需从外界输入数据,如计算5!;而,有些算法则需要输人数据,如计算n!,n的值是未知的,执行时需要从键盘输入n的值后,再计算。
么顾悉召队.乌气
(5)必须有一个或者多个输出。算法的实现是以得到计算结果为目的的,没有任何,输出的算法没有任何意义。
乡句1输入和分么

算法的描述方法

算法的描述方法主要有如下几种。1,自然语言描述用自然语言(Natural Language)描述算法时,可使用汉语、英语和数学符号等,通俗,-
懂,比较符合人们的日常思维习惯,但描述文字显得冗长,在内容表达上容易引起理解:的歧义,不易直接转化为程序,所以一般适用于算法较为简单的情况。

2.流程图描述
流程图(Flow Chart)是一个描述程序的控制流程和指令执行情况的有向图,它是程的一种比较直观的表示形式。美国国家标准化协会(ANSI)规定了如图5-1所示的符作为常用的流程图符号。用传统流程图描述算法的优点是流程图可直接转化为程序,象直观,各种操作一目了然,不会产生歧义,易于理解和发现算法设计中存在的错误;缺点是所占篇幅较大,允许使用流程线,使用者可使流程任意转向,有可能造成程序阅!印修改上的困难。

3. NS流程图描述
NS结构化流程图是由美国学者I.Nassi和B, Schneiderman于1973年提出的,NS图从而进行确立。优点是形象，直观，节省篇幅，尤其适合结构化程序设计。

4. 伪码描述

伪码是介于自然语言和计算机语言之间的一种代码，其最大的优点是与计算机语言比较接近，容易转化为计算机程序。书写无固定的格式和规范，十分灵活。

关系运算符及使用方式（>、>=、<、<=、==、！=）

运算符	含义
>	大于
>=	大于或等于
<	小于
<=	小于或等于
==	等于、优先级低
！=	不等于，优先级低

关系运算符用于比较两个数值之间的关系，例如：a>3为关系表达式，大于号为关系运算符，当表达式成立时，“a>3”的值为“真”，当“a>3”不成立时，“a>3”的值为“假”。

其中应当注意的是关系表达式返回值为整型值，而不是布尔型。表达式为真时返回值为1，表达式为假时返回值为0。

两种结构，一种称为顺序结构，也就是从上到下依次执行的过程称为顺序执行。

还有一种是选择结构，或者称为分支结构，需要进行判断，判断成功走一个分支，不成功走另一个分支。

两种错误，编译时错误是语法错误。

逻辑错误是运行时错误。

条件语句的内容：

单分支选择结构，只有if

双分支选择结构，if和else

多分支选择结构，if和else下面再进行嵌套

条件运算符和条件表达式

a<b?a:b;

成立选a，不成立选b

花括号下的内容称为语句块的内容

浮点数并非真正意义上的实数，只是其在某种范围内的近似，因此也是用近似的方法将实数与0进行比较

disc<= eps

eps设置为10的-8次方的大小。当小于一个那么小的数的时候就可以认为这个数据为0了。

程序运行过程中，根据不同的条件去执行不同的语句，常见的判断语句有if….else 、switch等。

（1） if语句的一般形式

```
If（表达式） 语句1
Else if（表达式） 语句2
….
Else 语句 n
```

（2）switch语句的一般形式

```
Switch(表达式)
{
        Case 常量表达式1： 语句1；break;
    Case 常量表达式2:  语句2；break;
    Case 常量表达式3:  语句3；break;
  …….
        Default : 语句 n+1; break;
}
```

对于switch语句的使用

switch语句的内容只能是“char” 和int类型的内容

char类型的可以是一些符号或者指定的字符的结构，而不能是其他的一些结构类型

常量和case中间至少有一个空格，常量后面的是冒号，常量的类型应该与switch后括号内表达式的类型相一致

break才会跳出，default作为默认的输入和输出结构即可。其不需要输入任何的待定结果项目

在存在除数的问题的时候需要考虑一个比较关键的问题，除数是否为0。也就是说是否存在除零

```
#include<stdio.h>
int main(){
    int b = 0;
    switch(b)
```

```
        {
        case 2:
                printf("h2\n");
                break;
        default:
                printf("。。。。。。。。\n");
        }
        return 0;
}
```

//exit函数的作用

//其的作用是终止整个程序的执行，强制返回操作系统

//当值为0或者EXIT_SUCCESS是正常的出!=0或者为EXIT_FAILURE表示程序出错后退出

//在数学上成立的表达式在C语言中不一定是正确的

问题精讲：短路与和短路或

```
/*
1.  或逻辑短路
#include <stdio.h>
int main()
{
        int a=5,b=6,c=7,d=8,m=2,n=2;
        (m=a<b)||(n=c>d);
        printf("%d\t%d",m,n);
        return 0;
}
```

输出的结果为1,2.为什么呢，因为a<b，m=1,这个“或”逻辑就被“短路”掉了，后面的语句就没被执行，所以n还是等于原先的2

```
2.  与逻辑短路
#include <stdio.h>
int main()
{
        int a=5,b=6,c=7,d=8,m=2,n=2;
```

```
        (m=a>b)&&(n=c>d);
        printf("%d\t%d",m,n);
        return 0;
}
```

输出的结果为0,2。因为a>b为0，m=0，整个“与”逻辑判断就为“假”，所以后面的“c>d”就被短路掉了，所以n还是等于原先的2

总结：

>>>>>“或”逻辑前面为1(true)就会发生短路。短路的时候后面的内容就不再进行判断了
>>>>>“与”逻辑前面为0(true)就会发生短路
>>>>> !的高于&&高于|

//计算某一年是否是闰年

```
#include <stdio.h>
int main()
{
        int a=5;
                int year;
                printf("please put need year:");
                scanf("%d",&year);
                if((year%4 == 0)&&(year %100 != 0)||(year %400 == 0))
                {
                        printf("%d是闰年\n",year);
                }
                else
                {
                        printf("%d不是闰年\n",year);
                }
                return 0;
}
```

逻辑运算符及使用方式（&&、||、！）

运算符	含义	举例	结果
&&	逻辑与	a&&b	a,b都为真则结果为真，否则为假
||	逻辑或	a||b	a,b至少有一个为真则结果为真，否则为假
!	逻辑非	!a	当a为真则结果为假，当a为假则结果为真

其中应当注意逻辑或，例如a||b，当a为真时，C语言中直接跳过对b的判断，其返回值为“真”。

运算符&&和||都具有

当一个表达式包括几种运算符时，则以运算符的优先级对表达式进行运算，表达式的优先级如下：

优先级	运算符类型	说明
1	初等运算符，从右向左	()、[]、->、.
2	单目运算符	！、~、++、--、*(指针运算符）、&(取地址运算符)
3	算术运算符	先乘除后加减
4	关系运算符	>、>=、<、<=、==、！=
5	逻辑运算符	&&、||
6	条件运算符	三目运算符，例如？：
7	赋值运算符	=
8	逗号运算符，从右向左	,

软件测试

程序测试只能证明程序有错，而不能证明程序无错

程序测试的目的就是为了尽可能多的发现程序中的错误

为了测试设计的数据称为测试用例

黑盒测试时功能测试

白盒测试是结构测试

一般测试的时候将白盒测试和黑盒测试相互结合的使用：选择有限数量的重要路径进行白盒测试，对重要的功能需求进行黑盒测试。

在测试其的一些边界的时候还可以使用边界测试，输入临界的数据或者临界的点。

进行按位运算：

优先级从上到下依次从高到低

~非

>>右移　再向外或者向右移动都会损失一定的数据

<<左移
&与
^异或
|或

快速幂 + 快速幂取模

1.最一般的求a的b次方

a^b

如果b是偶数

a^b=a^(b/2)*a^(b/2)

如果b是奇数

a^b=a^(b/2)*a^(b/2)*a

这样的话目标就转化为求

a^(b/2)了

继续用上面的方法，直到b/2==1为止，就可以返回a了

可以看到b 每次都除以2，那么最多是log(b)次就到1了

所以复杂度是log(b)

算法可以用递归实现，也可以用循环实现。

非递归

```
int pow_mod(int d,int n,int m)
{

int ret=1;
 int tempd=d;
 while(n)
 {
```

```
  if(n&1){ret*=d;ret%=m;}
  d*=d;
  n>>=1;
}
 return ret;
}
```

3.快速幂取模

公式： (a^b)%c =(a%c)^b %c

再改进：

这个算法在时间复杂度上没有改进，仍为O(b)，不过已经好很多的，但是在c过大的条件下，还是很有可能超时，所以，我们推出以下的快速幂算法。

快速幂算法依赖于以下公式

有了上述两个公式后，我们可以得出以下的结论：

$$a^b \bmod c = \begin{cases} ((a^2)^{b/2} * a) \bmod c, & b \text{ 是奇数} \\ ((a^2)^{b/2}) \bmod c, & b \text{ 是偶数} \end{cases}$$

我们可以看到，我们把时间复杂度变成了O(b/2).当然，这样子治标不治本。

但我们可以看到，当我们令k = (a * a) mod c时，状态已经发生了变化，我们所要求的最终结果即为(k)b/2mod c而不是原来的ab mod c，所以我们发现这个过程是可以迭代下去的。当然，对于奇数的情形会多出一项a mod c，所以为了完成迭代，当b是奇数时，我们通过 ans = (ans * a) % c;来弥补多出来的这一项，此时剩余的部分就可以进行迭代了。 形如上式的迭代下去后，当b=0时，所有的因子都已经相乘，算法结束。于是便可以在O（log b）的时间内完成了。于是，有了最终的算法：快速幂算法。 改进：(k在变动)