第4章 百花争艳香满园—程序设计基础

4.1程序设计语言简史

4.1程序设计语言简史


百花争艳
Java
JavaScript
Python
C++
Haskell
PHP
Pascal
Objective-C
Perl
Ruby
C
Assembly
Scratch
Scheme
Clarion
Transact-SQL
Visual Basic
ABAP
Logo
Prolog
ColdFusion
Forth
PL/SQL
Ada
Lua
Yacc
TCL
Delphi
MATLAB
ActionScript
PostScript
FoxPro
OpenEdge ABL
Scala
F#
Erlang
C#
D
ML
COBOL
Dart
Max/MSP
cT
SAS
Haskell
Tcl
Groovy
Lisp
Mathematica
Swift
Fortran
Go
R
Q

4.1程序设计语言简史

● 最早的程序语言—纸带打孔

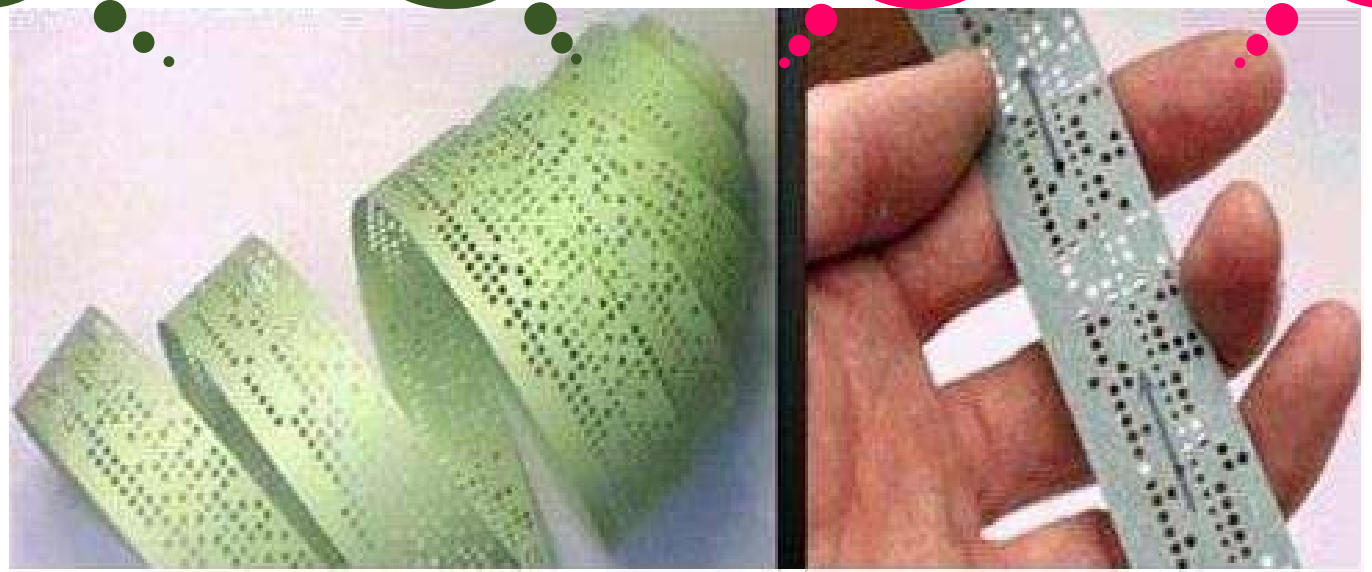
用“打孔/未打孔”的二值状态代表0和1，作为控制指令和数据存在物理介质上。是的，那个时代没有可视化界面，计算机真的只是“做计算的机器”，吞入纸带，完成运算，吐出纸带。

机器指令

执行效率高

编程效率低

容易出错



4.1程序设计语言简史

● 第一个符号语言—汇编语言

汇编语言（Assembly Language）是一种用于电子计算机、微处理器、微控制器或其他可编程器件的低级语言，亦称为**符号语言**。在汇编语言中，用**助记符**代替机器指令的操作码，用**地址符号或标号**代替指令或操作数的地址。

助记符
(相对而言)

不易出错
(相对而言)

```
_add_a_and_b:
    push    %ebx
    mov     %eax, [%esp+8]
    mov     %ebx, [%esp+12]
    add     %eax, %ebx
    pop     %ebx
    ret

_main:
    push    3
    push    2
    call    _add_a_and_b
    add     %esp, 8
    ret
```

求3+2的和

需要编译

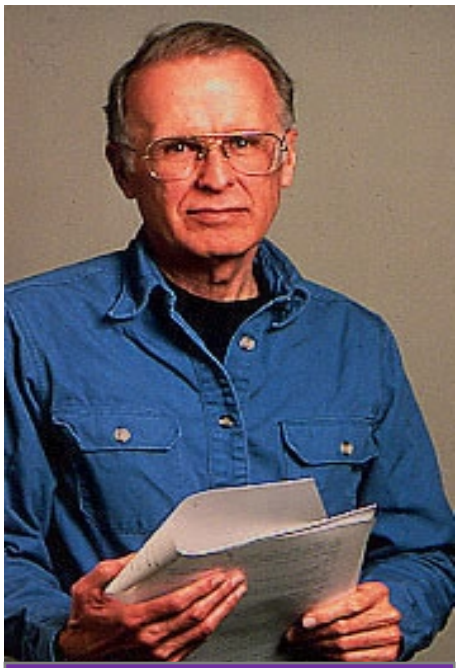
还是
过于复杂

4.1程序设计语言简史

● 第一个高级语言— Fortran

科学计算

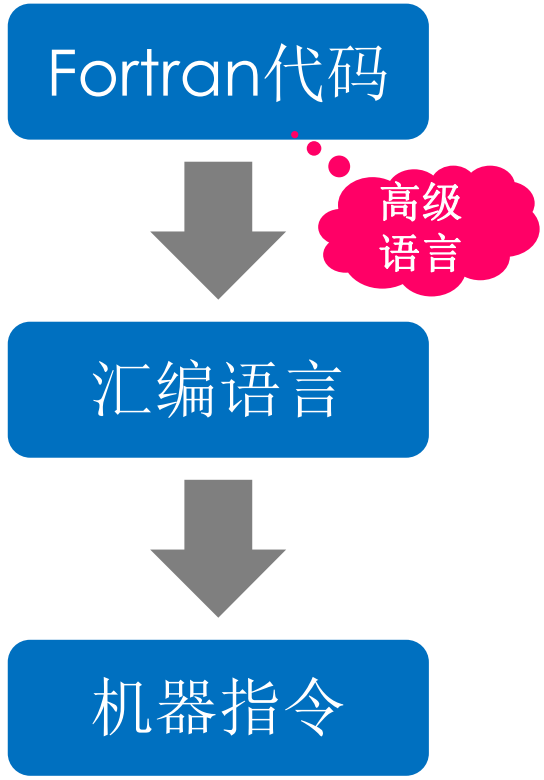
Fortran（Formula Translation）语言是为科学、工程问题或企事业管理中的那些能够用数学公式表达的问题而设计的，其数值计算的功能较强，是世界上第一个被正式推广使用的高级语言。



John Backus
Fortran之父

```
program main
    implicit none
    integer::i,j,N
    real*8::x(8)
    real*8::tmp
    x=[1,2,3,4,5,6,7,8]
    N=size(x)
    do i=1,N-1
        do j=i+1,N
            if(x(j)>x(i))then
                tmp=x(i)
                x(i)=x(j)
                x(j)=tmp
            end if
        end do
    end do
    do i=1,N
        print *, x(i)
    end do
end
```

冒泡排序Fortran程序



4.1程序设计语言简史

简单易学

● 初学者通用符号指令代码—BASIC

BASIC（Beginners’ All-purpose Symbolic Instruction Code）是“初学者通用符号指令代码”，一种设计给初学者使用的程序设计语言。BASIC是一种直译式的编程语言，在完成编写后不须经由编译及连结等手续即可执行，但如果需要单独执行时仍然需要将其建立成执行档。

J. Kemeny Thomas E.Kurtz

Basic之父

```
DATA 6
DATA 0.01,0.01,0.01,0.02,0.02,0.2,0.03,0.03,0.03,0.04,0.04,0.04,0.05,0.05,0.05,6.1,6.2,6.3
READ MO
DIM TO(MO+1),DO(MO),JO(MO)
FOR M=1 TO MO
    READ TO(M),DO(M),JO(M)
NEXT M

FOR M=1 TO MO
    PRINT "M=";M;",";TO(M);",";DO(M);",";JO(M)
NEXT M
END
```

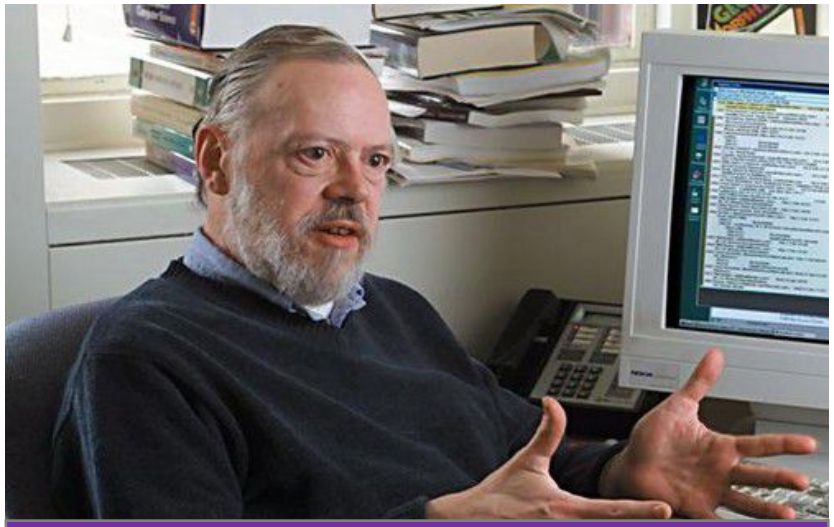
```
E:\basicProject>1.exe
M= 1,  0.01,  0.01,  0.01
M= 2,  0.02,  0.02,  0.2
M= 3,  0.03,  0.03,  0.03
M= 4,  0.04,  0.04,  0.04
M= 5,  0.05,  0.05,  0.05
M= 6,  6.1,  6.2,  6.3
```

4.1程序设计语言简史

适合
底层开发

● 程序设计入门必学语言—C语言

C语言是一门面向过程的、抽象化的，能以简易的方式编译、处理低级存储器、仅产生少量的机器码以及不需要任何运行环境支持便能运行的编程语言。C语言描述问题比汇编语言迅速、工作量小、可读性好、易于调试、修改和移植，而代码质量与汇编语言相当，广泛应用于底层开发。



Dennis M Ritchie
1983年ACM图灵奖
C语言之父、UNIX之父

```c
#include <stdio.h>
int main()
{
    int i,j,t,a[11];     //定义变量及数组为基本整型
    printf("请输入10个数：\n");
    for(i=1;i<11;i++)
        scanf("%d",&a[i]);     //从键盘中输入10个数
    for(i=1;i<10;i++)     //变量i代表比较的趟数
        for(j=1;j<11-i;j++)     //变量j代表每趟两两比较的次数
            if(a[j]>a[j+1])
            {
                t=a[j];     //产利用中间变量实现两值互换
                a[j]=a[j+1];
                a[j+1]=t;
            }
            printf("排序后的顺序是：\n");
            for(i=1;i<=10;i++)
                printf("%5d",a[i]);     //将冒泡排序后的顺序输出
        printf("\n");
    return 0;
}
```

UNIX是C的演示程序
（证明C语言有用）

4.1程序设计语言简史

C中加入“类”

● 第一个面向对象开发语言—C++

C++是C语言的继承，它既可以进行C语言的过程化程序设计，又可以进行以抽象数据类型为特点的基于对象的程序设计，还可以进行以继承和多态为特点的面向对象的程序设计，不仅拥有计算机高效运行的实用性特征，同时还致力于提高大规模程序的编程质量与程序设计语言的问题描述能力。

Bjarne Stroustrup
C++语言之父

```
#include <iostream>
    using namespace std;
    int main()
    {
        int x,y;
        for(x = 1;x <= 9; x++)
        {
            for(y = 9;y >= x; y--)
            {
                cout<<(9-y+1)<<"*"<<x<<"="<<x*(9-y+1)<<"\t";
            }
            cout<<endl;
        }
        return 0;
    }
```

4.1程序设计语言简史

不要{ }

- 具备大数据处理能力—Python

Python的设计目标之一是让代码具备高度的可阅读性。它设计时尽量使用其它语言经常使用的标点符号和英文单字，让代码看起来整洁美观。Python开发者有意让违反了缩进规则的程序不能通过编译，以此来强制程序员养成良好的编程习惯。

Guido van Rossum
Python语言之父

```
def save_candiwords(self, pmi_dict, candipos_path, candineg_path):
    def get_tag(word):
        if word:
            return [item.flag for item in pseg.cut(word)][0]
        else:
            return 'x'
    pos_dict = dict()
    neg_dict = dict()
    f_neg = open(candineg_path, 'w+')
    f_pos = open(candipos_path, 'w+')

    for word, word_score in pmi_dict.items():
        if word_score > 0:
            pos_dict[word] = word_score
        else:
            neg_dict[word] = abs(word_score)

    for word, pmi in sorted(pos_dict.items(), key=lambda asd:asd[1], reverse=True):
        f_pos.write(word + ',' + str(pmi) + ',' + 'pos' + ',' + str(len(word)) + ',' + get_tag(word) + '\n')
    for word, pmi in sorted(neg_dict.items(), key=lambda asd:asd[1], reverse=True):
        f_neg.write(word + ',' + str(pmi) + ',' + 'neg' + ',' + str(len(word)) + ',' + get_tag(word) + '\n')
    f_neg.close()
    f_pos.close()
    return
```

4.1程序设计语言简史

取消指针

● 纯面向对象开发语言—Java

Java是一门面向对象编程语言，不仅吸收了C++语言的各种优点，还摒弃了C++里难以理解的**多继承**、**指针**等概念，具有简单性、面向对象、分布式、健壮性、安全性、平台独立与可移植性、多线程、动态性等特点。

James Gosling
Java语言之父

```
package pp;
import java.awt.*;

public class pp{

    public pp() {
        Frame app = new Frame("DrawBall2");
        //监视关闭窗体事件
        app.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
        app.setLocation(100, 100);
        MyPanel drawB = new MyPanel(); //实例化MyPanel对象
        app.add(drawB, BorderLayout.CENTER); //在窗体中间显示画版

        app.pack();//运行app中的pack方法,动态调整frame的大小，使frame中的组件都可见
        app.setVisible(true);
        drawB.gameStart (); //运行app中的gameStart方法
    }
```

关闭

最流行的开发语言

4.1程序设计语言简史

开源

Web开发语言—PHP

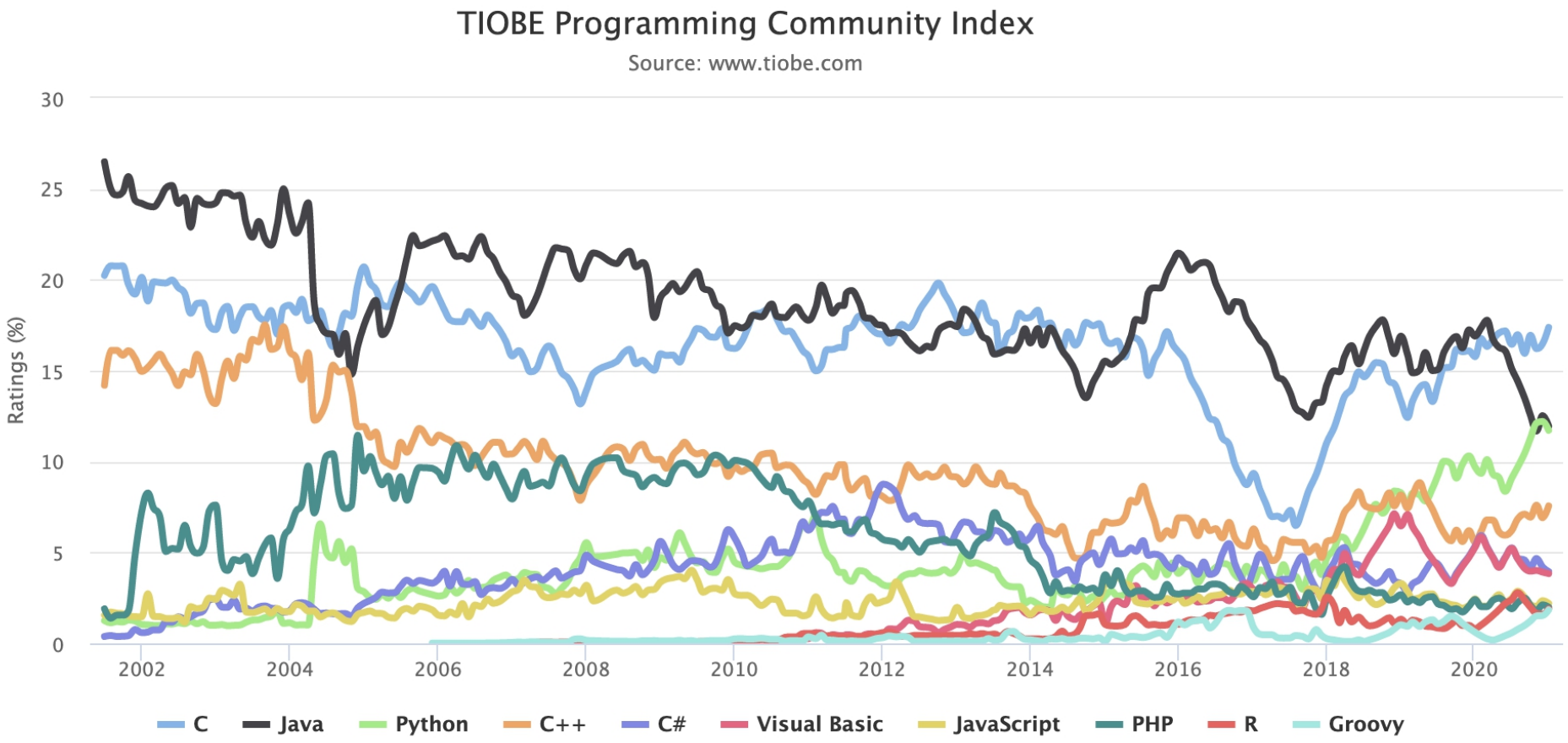
PHP（PHP: Hypertext Preprocessor）即“超文本预处理器”，是在服务器端执行的脚本语言，尤其适用于Web开发并可嵌入HTML中。PHP语法学习了C语言，吸纳Java和Perl多个语言的特色发展出自己的特色语法，当初创建的主要目标是让开发人员快速编写出优质的web网站。

Rasmus Lerdorf
PHP语言之父

```php
<?php
header("Content-type: text/html; charset=utf-8"); //设置编码格式
    if(isset($_POST['sub'])){           //判断是否提交
        if(!empty($_POST['phone'])){    //判断是否为空
                $phone = $_POST['phone']; //电话号码
                //echo $phone;

                $ch = curl_init();
                curl_setopt($ch, CURLOPT_URL, "http://api.weimi.cc/2/sms/send.html"); //接口地址
                curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
                curl_setopt($ch, CURLOPT_POST, TRUE);
                /*
                短信接口二，触发类模板短信接口，可以设置动态参数变量。适合：验证码、订单短信等。
                1、设定微米账号的接口UID和接口密码
                2、传入目标手机号码，多个以“,”分隔，一次性调用最多100个号码，示例：139********,138********
                3、短信模板cid，通过微米后台创建，由在线客服审核。必须设置好短信签名，签名规范：
                    1）、模板内容一定要带签名，签名放在模板内容的最前面；
                    2）、签名格式：【***】，签名内容为三个汉字以上（包括三个）；
                    3）、短信内容不允许双签名，即短信内容里只有一个“【】”。
                4、传入模板参数。短信模板内容示例：
                    【微米】您的验证码是：%P%，%P%分钟内有效。如非您本人操作，可忽略本消息。
                    传入两个参数：
                    p1：610912
                    p2：3
                    最终发送内容：
                    【微米】您的验证码是：610912，3分钟内有效。如非您本人操作，可忽略本消息。
                */
                curl_setopt($ch, CURLOPT_POSTFIELDS, "uid=AnnQfuovlED9&pas=cbx7x89r&mob=$phone
                    &cid=d9Z7uIKCYqhQ&p1=cp666&p2=3&type=json");
                $res = curl_exec( $ch );
                curl_close( $ch );
                echo $res;
                /*
                注意：以上参数传入时不包括“<>”符号
                */
        }else{
            echo "<script>alert('数据不能为空')</script>";
        }
    }
```

4.1程序设计语言简史



TIOBE 编程语言排行榜

https://www.tiobe.com/

4.1程序设计语言简史

几乎每个程序员都认为自己擅长的是最好的语言

语言之父们都非常有个性

“世界上最好的语言”之争

4.1程序设计语言简史

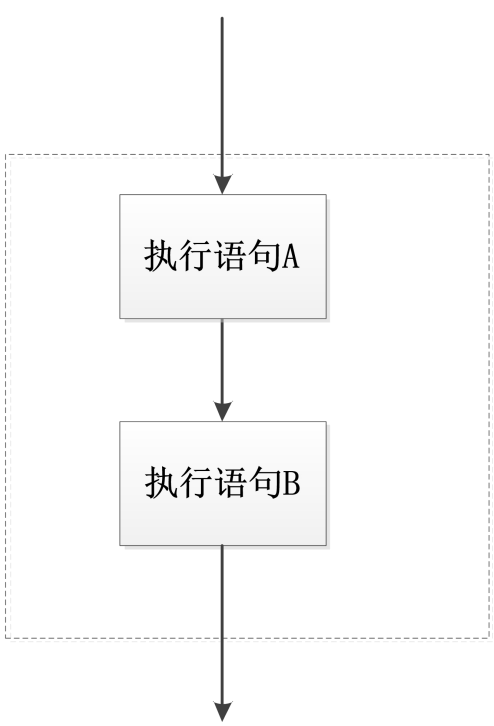
编程语言
优劣比较

内鬼Ayu bilibili

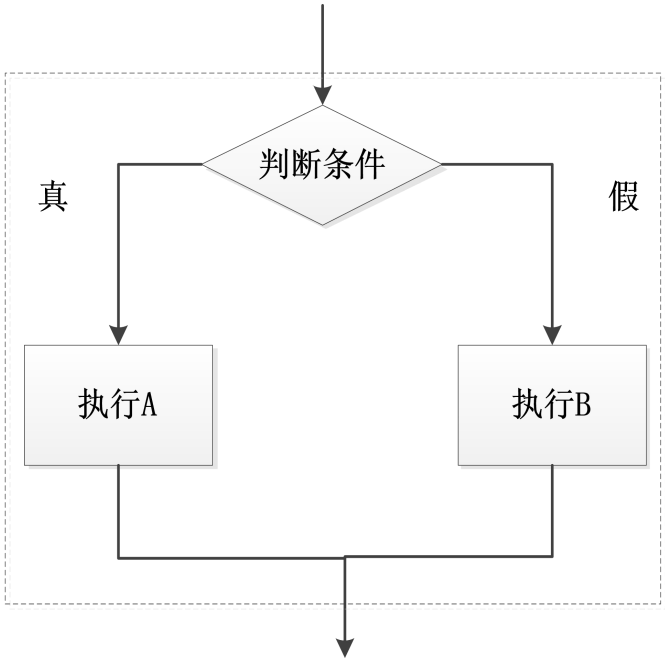
本期聊的语言我都用来做过项目
大部分都做过大型项目

4.2程序设计基本逻辑

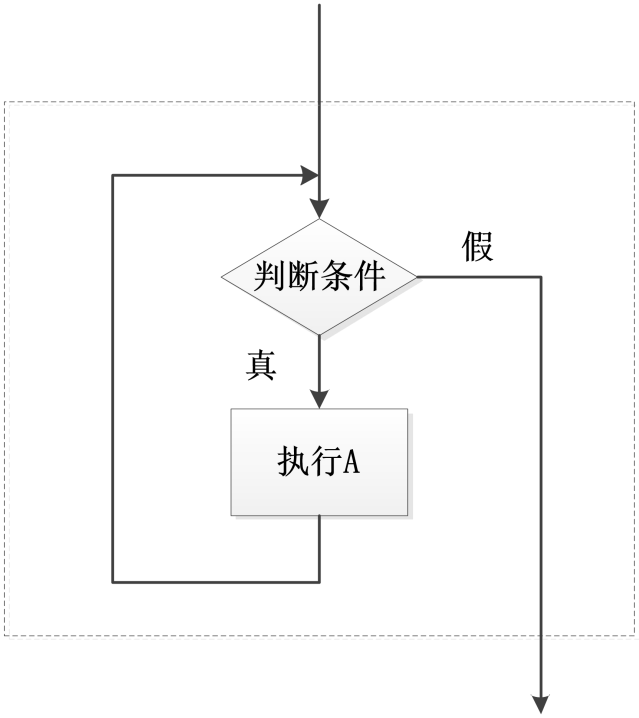
从程序流程的角度来看，程序由三种基本结构组成，分别是**顺序结构**、**分支结构**和**循环结构**。由这三种基本结构可以组合出任意复杂的程序。换句话说，任何一个结构程序都可以由这三种基本控制结构来表示。



顺序结构



选择结构

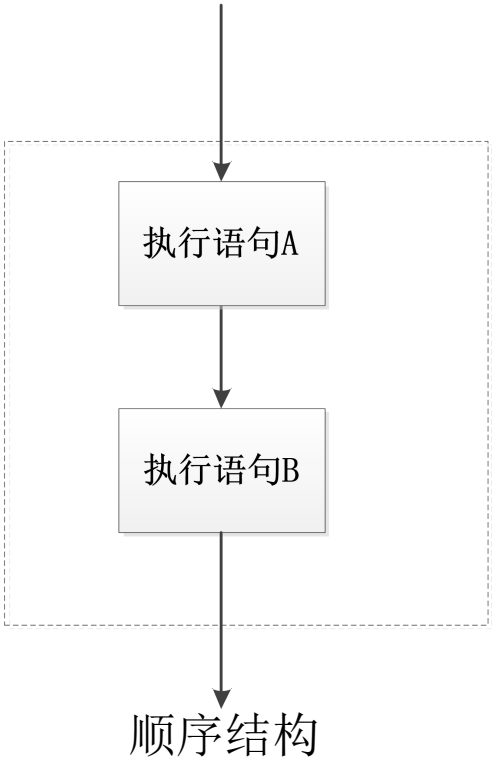


循环结构

4.2程序设计基本逻辑

● 顺序结构

顺序结构是最基本、最常用的结构。顺序结构就是按照程序语句的书写顺序一条一条地执行，并且每个语句都会被执行到。如下图所示，这是一个顺序结构，程序会先执行语句A，然后执行语句B。

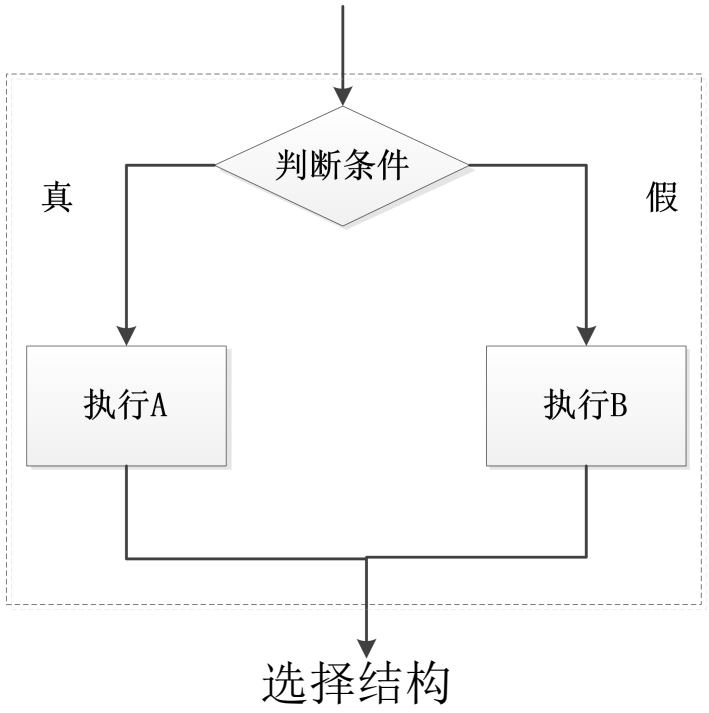


顺序结构

4.2程序设计基本逻辑

● 选择结构

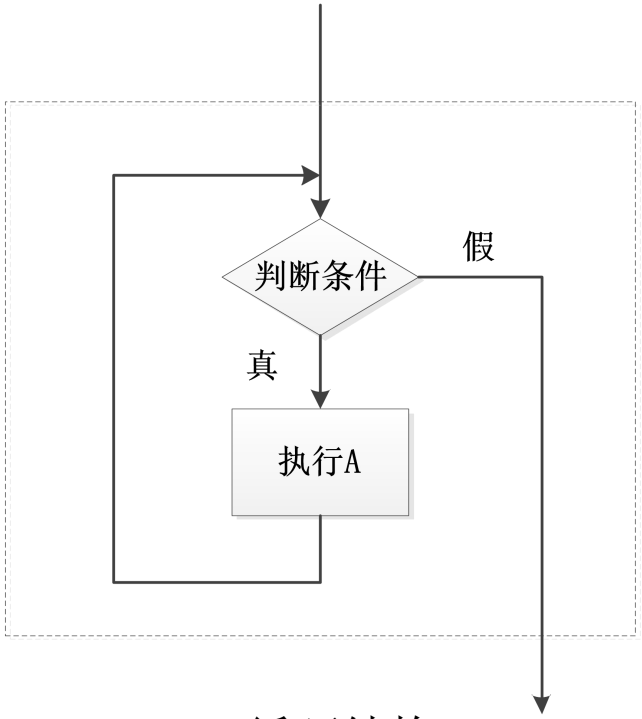
选择结构表示程序的处理需要根据某个特定的逻辑条件的成立与否，选择其中的一个分支执行，每个分支完成不同的功能。如图所示，程序开始会判断条件是真是假，如果条件为真，则执行语句A，如果条件为假，则执行语句B。



4.2程序设计基本逻辑

● 循环结构

用计算机解决实际问题时，常常需要处理重复的内容，循环结构又称为重复结构。根据给定的条件，判断是否需要重复执行相同的程序段，利用循环结构可以简化大量的代码。



循环结构

4.2程序设计基本逻辑

4.2程序设计基本逻辑

错误逻辑

```
#include "stdio.h"
#include "math.h"
main()
{
    int a,b,c;
    bool flag=true;              //是否为素数的标志
    for(n=3;n=<=100;n+=2)        //偶数肯定不是，所以n+=2
    {
        flag=true;               //必须再次赋值
        m=sqrt(n);               //整除的截止值
        for(i=2;i<=m;i++)
        {
            if(n%i==0)//n%i取余数
            {
                flag=false;//能整除，肯定不是
                break;     //跳出当前for循环
            }else{
                printf("%d     ",n); //输出素数
            }
        }
    }
}
```

能整除不是素数，否则就是？

4.2程序设计基本逻辑

正确逻辑

```
#include "stdio.h"
#include "math.h"
main()
{
    int a,b,c;
    bool flag=true;              //是否为素数的标志
    for(n=3;n=<=100;n+=2)  //偶数肯定不是，所以n+=2
    {
        flag=true;               //必须再次赋值
        m=sqrt(n);               //整除的截止值
        for(i=2;i<=m;i++)
        {
            if(n%i==0)//n%i取余数
            {
                flag=false;//能整除，肯定不是
                break;      //跳出当前for循环
            }
        }
        if(flag)
            printf("%d     ",n); //输出素数
    }
}
```

4.3数据结构和算法

早期的计算机主要应用于**数值计算**（方程、方程组），通过编写程序即可求解该类数值问题；随着计算机的应用领域不断扩展，逐渐开始应用到“**非数值计算**”领域，传统的求解思路，无法求解此类“非数值”问题，“数据结构”应运而生。

非数值问题

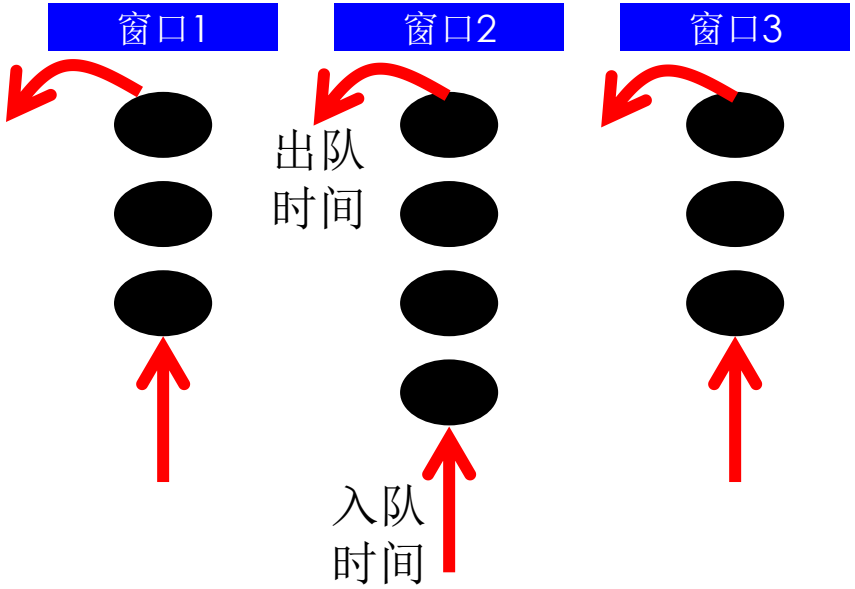
字符　字符串　文字　图形　图象　声音

星期		星期一	星期二	星期三	星期四	星期五	星期六
上午	一			民事诉讼法 韦留柱 XL-101	大学英语 李振伟 A-302	民事诉讼法 韦留柱 XL-101	
	二	商法总论 于庆生 XL-101	VF 董永亮 XL-309		商法总论（单周） 于庆生 XL-101	经济法概论（单周） 王晓慧 XL-301	
中午	三						
下午	四	大学英语 李振伟 A-302		大学体育		VF（单周） 董永亮 XL-309	
	五	经济法概论 王晓慧 XL-201			证据学 石利 XL-103	物权法 张河顺 XL-301	
晚上		美术鉴赏 赵倩 XL-301		物权法 张河顺 XL-301			

4.3数据结构和算法

队列

中午食堂需要开几个窗口，才不至于使排队时间过长?



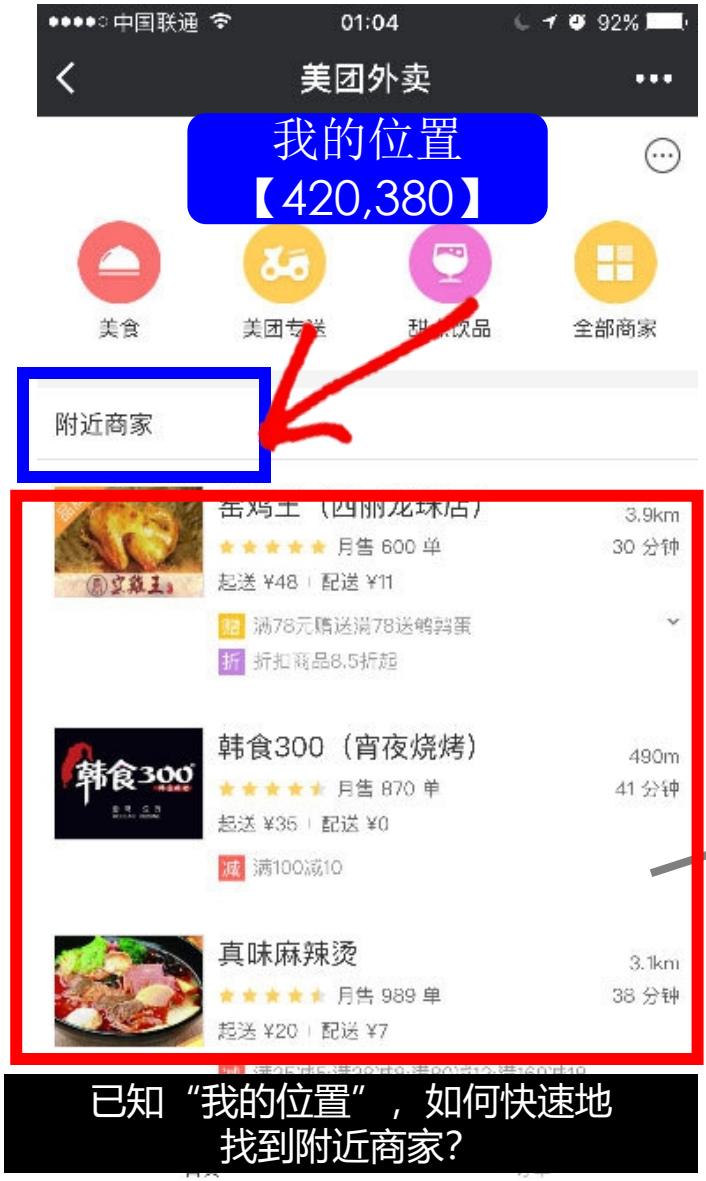
排队时间 = 出队时间-入队时间

超市收银通道数量?

银行窗口数量?

地铁闸机数量?

4.3数据结构和算法



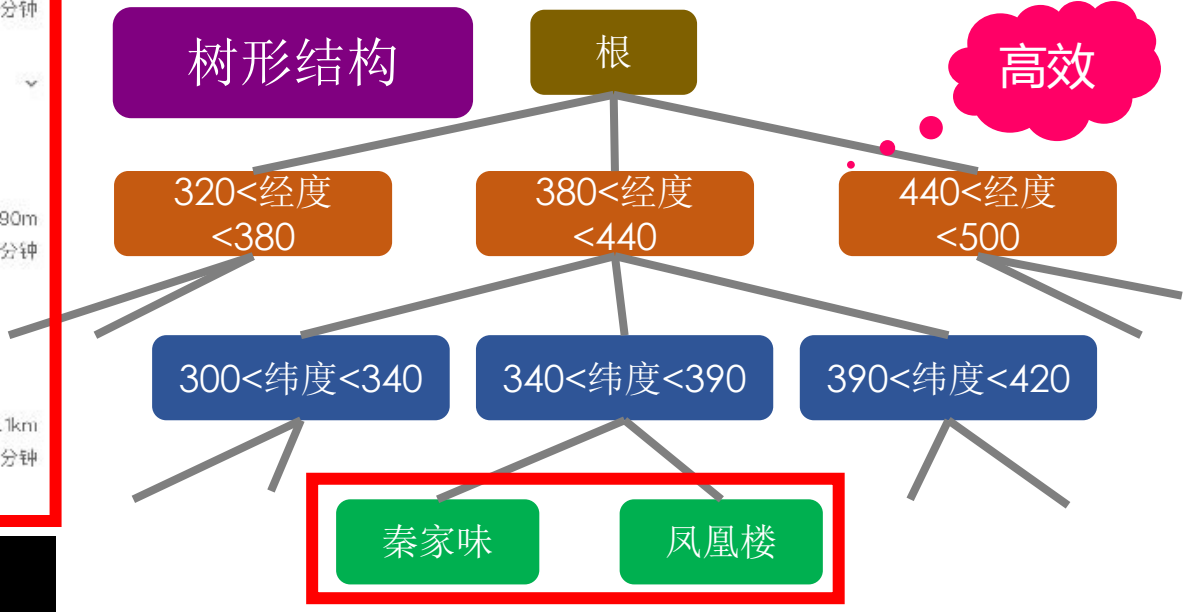
已知“我的位置”，如何快速地找到附近商家？

逐个比较
效率极低

商家ID	商家名称	商家位置（经纬度）
0335000011	秦家味	【430,550】
0335000012	蜜雪冰城	【200,362】
0335000013	凤凰楼	【428,380】
0335000014	万州烤鱼	【430,890】
……		
……		
0335010011	外婆铁锅焖面	【528,780】

10W+
商家

树形结构



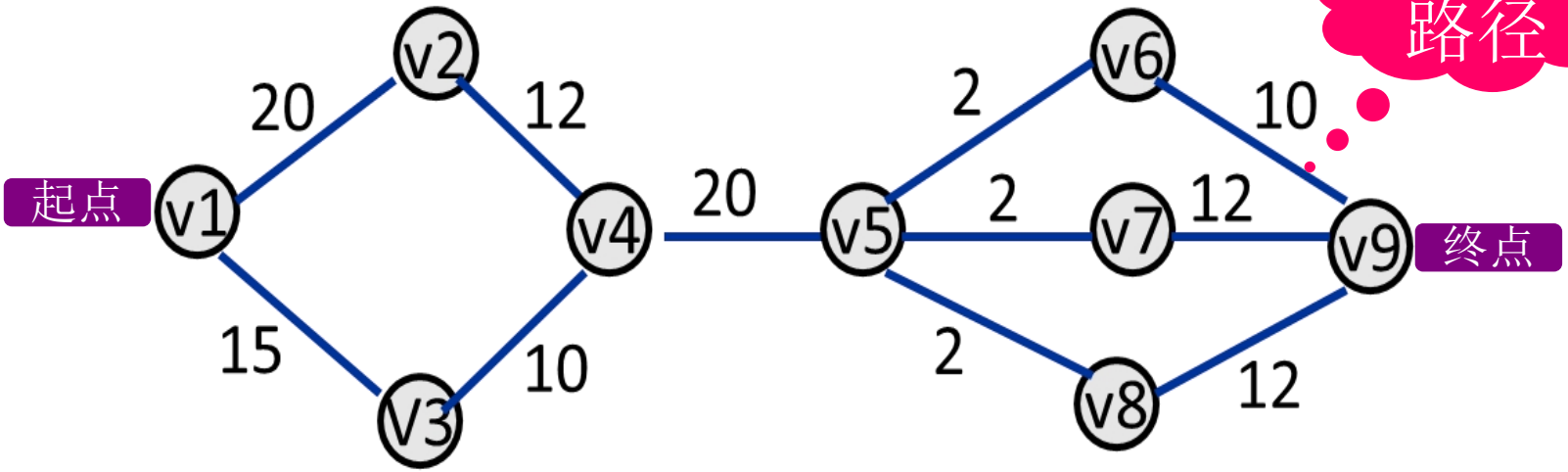
高效

4.3数据结构和算法

图形/网状

导航系统、通讯网、计算机网络、游戏等常要寻找两结点间最短路径。

从西区燕园食堂出发到东区图书馆，走哪条路最短？？



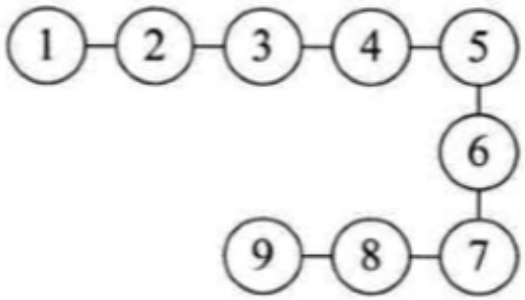
4.3数据结构和算法

综上，描述这类非数值问题的数学模型不再是数学方程，而是诸如表、树和图之类的数据结构。数据结构是一门研究**非数值计算**的程序设计问题中计算机的操作对象以及它们之间的关系和操作等等的学科。

- **线性结构**

队列

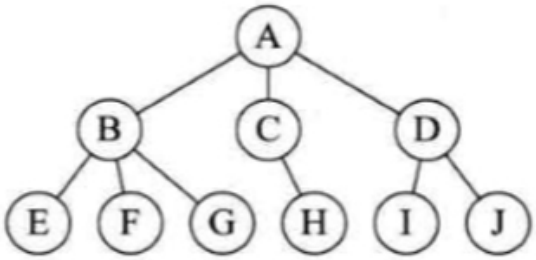
线性结构中的数据元素之间存在一对一的关系。



- **树形结构**

树

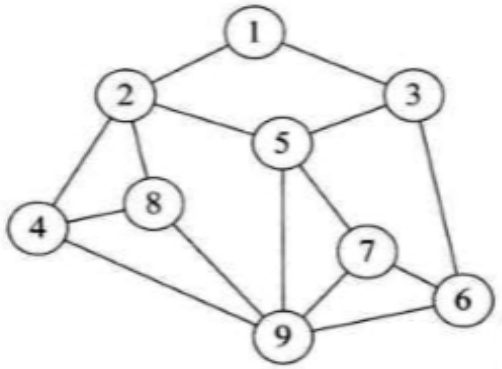
树形结构中的数据元素之间存在一对多的关系。



- **图状结构**

图

图状结构中的数据元素之间存在多对多的关系。



4.3数据结构和算法

- 树形结构的存储方式：二叉链表
- data表示数据域、lchild、rchild都是指针域，分别存放指向左孩子和右孩子的指针。


```
/*二叉树的二叉链表节点结构定义*/
typedef struct BiTNode   /* 结点结构 */
{
    TElemtype data;      /* 结点数据 */
    struct BiTNode *lchild, *rchild;          /* 左右孩子指针 */
}BiTNode,*BiTree;
```

4.3数据结构和算法

综上，描述这类非数值问题的数学模型不再是数学方程，而是诸如表、树和图之类的数据结构。数据结构是一门研究**非数值计算**的程序设计问题中计算机的操作对象以及它们之间的关系和操作等等的学科。

4.3数据结构和算法

- 算法

算法是对问题求解过程的一种描述，是为解决一个或一类问题给出的一个确定的、有限长的操作序列。

有穷性：算法中的操作步骤为有限个，且每个步骤都能在有限时间内完成。

确定性：组成算法的操作必须清晰无二义性。

可行性：算法中的所有操作都必须足够基本，都可以通过已经实现的基本操作运算有限次实现之。

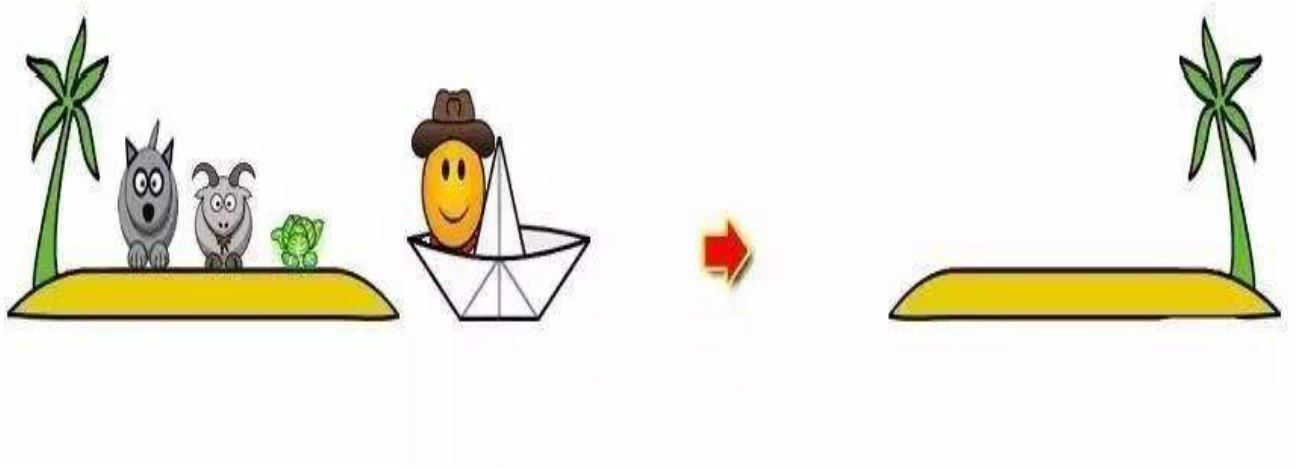
输入：作为算法加工对象的量值，通常体现为算法中的一组变量。一些算法的字面上可以没有输入，实际上已被嵌入算法之中。

输出：它是一组与输入有确定关系的量值，是算法进行信息加工后得到的结果，这种确定关系即为算法的功能。

4.3数据结构和算法

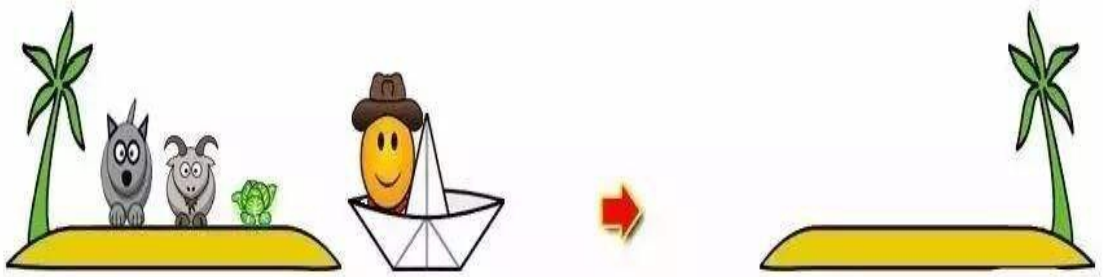
● 算法

对于初学者来说，算法可能是第一次听到过，但其实我们很早就已经接触过算法了，算法一直在我们身边。相信大家都听说过农夫过河的问题，一个农夫带着一只狼，一只羊和一颗白菜过河，河边只有一条船，由于船小，农夫一次只能带其中的一样过河，如图。如无人看管，狼要吃羊，羊要吃菜。问农夫如何安排过河，才能使得羊、菜都安然无恙？



4.3数据结构和算法

- 算法



（1）农夫带着羊渡过河去
（2）农夫划船回来
（3）农夫带着菜渡过河去
（4）农夫带着羊划船回来
（5）农夫带着狼渡过河去
（6）农夫划船来
（7）农夫带着羊渡过河

4.3数据结构和算法

● 背包问题—贪心算法

有一个背包，最多放M kg的物体（物体大小不限）；

有n个物体，每个物体的重量为Wi，

每个物体完全放入背包后可获得收益Pi。

问：如何放置能获得最大的收益？

算法思路：将物品按照单位重量价值进行排序（从大到小），将尽可能多的单位重量价值最高的物品装入背包，若将这种物品全部装入背包后，背包还有多余容量，则选择单位重量价值次高的并尽可能多地装入背包。如果最后一件物品无法全部装入，则计算可以装入的比例，然后按比例装入。

4.3数据结构和算法

- **使用不同的算法计算**1+2+3+ • • • +100=?

```
//算法一
int i, sum = 0, n = 100;
for(i = 1; i <= n; i++)
{
    sum = sum + i;
}
printf("%d ",sum);
```

```
//算法二
int sum = 0, n = 100;
sum = (1 + n) * n / 2;
print("%d ",sum);
```

高斯解释：

sum = 1 + 2 + 3 + …+ 100

sum = 100 + 99 + 98 + …+ 1

2*sum = 101 +101+101+ …+ 101

共100个

所以sum = 5050

相同的问题，使用不同的算法，有时候会事半功倍！

4.3数据结构和算法

算法的评价标准

- 正确性：算法的正确性是指算法至少应该具有输入、输出和加工处理无歧义性、能正确反映问题的需求、能够得到问题的正确答案。
- 可读性：为了便于理解、测试和修改算法，算法应该具有良好的可读性。
- 健壮性：一个好的算法还应该能对输入数据不合法的情况做合适的处理，算法中拥有对输入数据、打开文件、读取文件记录、分配内存空间等操作的结果检测，并通过与用户对话的形式做出相应的处理选择。
- 时间效率高和存储量低：算法的时间与空间效率是指将算法变换为程序后，该程序在计算机上运行时所花费的时间少及所占据空间的度量低。

4.3数据结构和算法

- 算法效率的度量

✓事后统计的方法

•优点：精确

•缺点：需要先写出程序，软硬件环境的影响

没有代表性！

✓事前分析估算的方法

采用以求解问题的基本操作（原操作）的执行次数作为算法时间的度量。

允许不准确

4.3数据结构和算法

- 算法效率的度量

```
//算法一
int i, sum = 0, n = 100;          /* 执行 1 次 */
for(i = 1; i <= n; i++)           /* 执行 n+1 次 */
{
    sum = sum + i;                /* 执行 n 次 */
}
printf("%d ",sum);                /* 执行 1 次 */
```

共执行1+（n+1）+ n +1= 2n+3 次

```
//算法二
int sum = 0, n = 100;             /* 执行 1 次 */
sum = (1 + n) * n / 2;            /* 执行 1 次 */
print("%d ",sum);                 /* 执行 1 次 */
```

共执行1+1+1= 3 次

4.3数据结构和算法

- 算法效率的度量

```
For ( i = 1; i<=n; i++ )
    For (j = 1; j<=n; j++ )  {
        c[ i ][ j ] = 0 ;
        For (k = 1; k<= n; k++ )
            c[ i ][ j ] += a[ i ][ k ] * b[ k ] [ j ]
    }
```

原操作

执行次数为 n^3

- 矩阵相乘算法执行时间与n^3成正比
- 算法的**时间复杂度**为O (n3)

4.3数据结构和算法

- 算法效率的度量

以下六种计算算法时间的多项式是最常用的。其关系为：

$$O(1)<O(\log^n)<O(n)<O(n\log^n)<O(n^2)<O(n^3)$$

指数时间的关系为：

$$O(2^n)<O(n!)<O(n^n)$$

因此，只要有人能将现有指数时间算法中的任何一个算法化简为多项式时间算法，那就取得了一个伟大的成就。

4.3数据结构和算法

- 算法效率的度量

编程实现求1~n之间所有整数的和。

```
int getSum(int n)
{
    int sum = 0;
    for(int i=1;i<=n;i++)
    {
        sum = sum+i;
    }
    return sum;
}
```

O(n)

对

```
int getSum(int n)
{
    int sum = 0;
    sum = (1+n)*n/2;
    return sum;
}
```

O(1)

好

4.3数据结构和算法

- 算法存储空间的度量

一个上机执行的程序除了需要存储空间来寄存本身所用指令、常数、变量和输入数据外，也需要一些对数据进行操作的工作单元和存储一些为实现计算所需信息的**辅助空间**。

通常，只有完成同一功能的几个算法之间才具有可比性，这些输入数据所占用的空间不用进行比较，因此只需**比较那些辅助的或附加的存储空间**。

空间复杂度
(Space Complexity)

4.3数据结构和算法

数据结构+算法 = 程序设计

牺牲空间复杂度，降低时间复杂度

内存空间越来越大，价格也越来越低
算法效率成为关键！

最新研究进展

序号	刊物名称	刊物全称	地址
1	PLDI	ACM SIGPLAN Symposium on Programming Language Design & Implementation	http://dblp.uni-trier.de/db/conf/pldi/
2	POPL	ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages	http://dblp.uni-trier.de/db/conf/popl/
3	OOPLSA	Conference on Object-Oriented Programming Systems, Languages,and Applications	http://dblp.uni-trier.de/db/conf/oopsla/