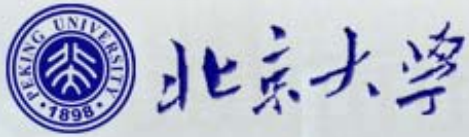
《计算概论A》课程 程序设计部分

程设那些事儿

李　戈

北京大学 信息科学技术学院 软件研究所

2010年12月31日



议题1：一些零散的小问题

先看一段程序

```
#include<iostream>
using namespace std;
int main( )
{
    int a = 1, b = 5;
    if(a/b==0.2)
        cout<<"right!"<<endl;
    else
        cout<<"Why?"<<endl;
    return 0;
}
```

再看一段程序

```
#include<iostream>
using namespace std;
int main( )
{
    int a = 1, b = 5;
    if(double(a)/b==0.2)
        cout<<"right!"<<endl;
    else
        cout<<"Why?"<<endl;
    return 0;
}
```

注意之一

- 不要:
  - 将浮点变量用“==”或“!=”与数字比较
- 要:
  - 设法转化成“>=”或“<=”形式。
- 例如:

对与浮点数x，可以将

```
if (x == 0.0) // 隐含错误的比较
```

转化为

```
if ( fabs(x) <= E-5)
```

先看一段程序

```
#include<iostream>
using namespace std;
int main()
{
    int sum=0, i;
    do{
        sum++;
        cout<<“Continue or Quit? (1，0)";
        cin>>i;
    }while(i != 0);
    return 0;
}
```

（2）再看一段程序

```
#include<iostream>
using namespace std;
int main()
{
    int sum = 0, i;
    do {
        sum ++;
        cout<< "continue? (1，0)";
        cin >> i;
    } while( i == 1);
    return 0;
}
```

注意之二

- 结论:
  - 为了避免死循环的产生，应该让循环的继续的条件是严格的，而退出循环的条件是宽松的。
- 例如:

```
do {
    sum ++;
    cout<< "continue? (1，0)";
    cin >> i;
} while( i == 1); // 优于while(i != 0)
```

注意之三：系统也糊涂

```
#include<iostream>
using namespace std;
int main()
{
    const int a = 2.6;
    cout << a;
    cout << endl;
    return 0;
}
```

输出：

1895105668

很明显，这是系统Bug!

不知道微软是怎么进行测试的？！

议题2：什么样的程序更好?

整数奇偶排序

- Description

  输入10个整数，彼此以空格分隔

  重新排序以后输出(也按空格分隔)，要求:

  1.先输出其中的奇数,并按从大到小排列;

  2.然后输出其中的偶数,并按从小到大排列。

- Input

  任意排序的10个整数（0～100），彼此以空格分隔

- Output

  按照要求排序后输出，由空格分隔

```
#include<iostream>
using namespace std;
int main(){
    int a[10],b[10],c[10],i,j,k,m,n,t;
    while(cin>>a[0]>>a[1]>>a[2]>>a[3]>>a[4]>>a[5]>>a[6]>>a[7]>>a[8]>>a[9]){
        j=0;
        k=0;
        for(i=0;i<=9;i++){
            if(a[i]%2!=0)b[j++]=a[i];
            if(a[i]%2==0)c[k++]=a[i];
        }
        for(i=0;i<j;i++){
            for(m=j-1;m>=1+i;m--){
                if(b[m]>b[m-1]){
                    t=b[m-1];
                    b[m-1]=b[m];
                    b[m]=t;
                }
            }
        }
```

```
        for(i=0;i<k-1;i++){
                for(n=0;n<k-1-i;n++){
                        if(c[n]>c[n+1]){
                                t=c[n];
                                c[n]=c[n+1];
                                c[n+1]=t;
                        }
                }
        }
        for(i=0;i<j;i++){
                cout<<b[i]<<" ";
        }
        for(i=0;i<k;i++){
                cout<<c[i]<<" ";
        }
        cout<<endl;
    }
    return 0;
}
```

```
#include <iostream>
using namespace std;
int main( ){
    int a[10], i, j;
    while (cin>>a[0]>>a[1]>>a[2]>>a[3]>>a[4]>>a[5]>>a[6]>>a[7]>>a[8]>>a[9]){
        for (i=99; i>=1; i=i-2){
            for (j=0; j<=9; j++){
                if (a[j]==i)
                    cout<<a[j]<<" ";
            }
        }
        for (i=0; i<=100; i=i+2){
            for (j=0; j<=9; j++){
                if (a[j]==i)
                    cout<<a[j]<<" ";
            }
        }
        cout<<endl;
    }
    return 0;
}
```

议题3：程序应写成什么样?

写程序注意事项

- 变量命名
  - 匈牙利命名法 or 自己的命名方法;
- 注释！注释！注释！
  - 变量定义后
  - 函数定义前
  - 重要的逻辑判断之后
  - 输入输出之后
- 缩进!

写程序注意事项

- 最容易犯的错误
  - if (x = 1)
  - int sum;

    for(i = 0; i < 100; i++)

        sum++;
  - cout<<sum;
  - int a[100][100]

    … a[i][j]…

谨慎！

议题4：几个例题

字符串处理举例

- 问题:
  - 读入两个字符串a和b，判断a是不是b的子串，如果是，计算a在b中出现了几次。
  - 例如:

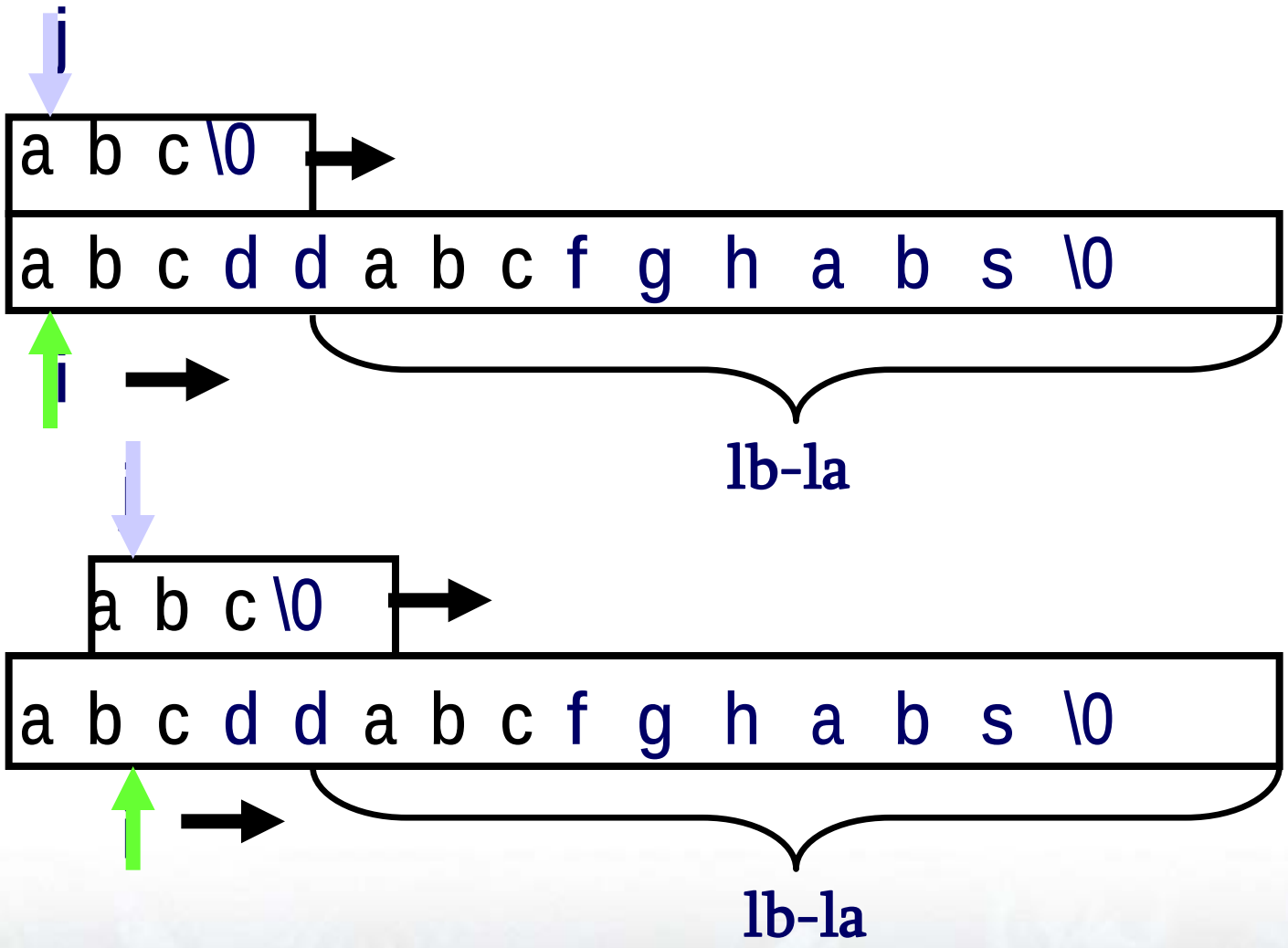
    a = “aba”;　　b = “abababab”;

    则 a 在 b 中出现了 2 次;

字符串处理举例

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
        char a[100], b[100];
        int search(char a[], char b[]);
        cin>>a>>b;
        cout<<search(a,b)<<endl;
        return 0;
}
```

字符串处理举例（2）



```
int search(char a[], char b[]) {
        bool match;
        int la=strlen(a), lb=strlen(b), count=0;
        for(int i = 0; i <= lb-la; i++){
                match = true;
                for(int j = 0; j < la; j++) {
                        if (a[j] != b[i+j]){
                                match = false;
                                break;
                        }
                }
                if(match) count++;
        }
        return count;
}
```

```
int search(char *a, char *b) {
        bool match;
        int la=strlen(a), lb=strlen(b), count=0;
        for(int i = 0; i <= lb-la; i++){
                match = true;
                for(int j = 0; j < la; j++) {
                        if (*(a+j)!= *(b+i+j)){
                                match = false;
                                break;
                        }
                }
                if(match) count++;
        }
        return count;
}
```

红与黑

- 问题
  - 有一间长方形的房子，地上铺了红色、黑色两种颜色的正方形瓷砖。你站在其中一块黑色的瓷砖上，只能向相邻的黑色瓷砖移动。请写一个程序，计算你总共能够到达多少块黑色的瓷砖。

红与黑

- 输入
  - 包括多个数据集合。每个数据集合的第一行是两个整数W和H，分别表示x方向和y方向瓷砖的数量。W和H都不超过20。在接下来的H行中，每行包括W个字符。每个字符表示一块瓷砖的颜色，规则如下

    1） ‘.’：黑色的瓷砖；

    2） ‘#’：白色的瓷砖；

    3） ‘@’：黑色的瓷砖，并且你站在这块瓷砖上。该字符在每个数据集合中唯一出现一次。

    当在一行中读入的是两个零时，表示输入结束。

红与黑

- 输出
  - 对每个数据集合，分别输出一行，显示你从初始位置出发能到达的瓷砖数(记数时包括初始位置的瓷砖)。

```
6 9
....#.
.....#
......
......
......
......
......
#@...#
.#..#.
11 9
.#.........
.#.########.
.#.#.....#.
.#.#.###.#.
.#.#..@#.#.
.#.#####.#.
.#.......#.
.#########.
...........
0 0
```

输入的方法之一

```
#include <iostream>
using namespace std;
int main()
{
    while(1)
    {
        cin>>H>>W;
        if(W == 0 || H == 0) break;
        ……
    }
    return 0;
}
```

输入的方法之二

```
#include <iostream>
using namespace std;
int main()
{
    while(cin>>H && cin>>W &&W != 0 &&H != 0)
    {

    }
    return 0;
}
```

思路一

- “试走法”
  - 从指定位置开始走；
    - 向上
    - 向下
    - 向左
    - 向右
  - 每到一处，只要符合条件，标记为'$'
  - 不符合条件什么都不做
  - “走完”后数$

```
11 6
..#..#..#..
..#..#..#..
..#..#..###
..#..#..#@.
..#..#..#..
..#..#..#..
```

```
void f(int x, int y)
{
 if(x<0 || x>=W || y<0 || y>=H || z[x][y] == '#' || z[x][y] == '$')
        // 如果走出矩阵范围
        return;
 else
 {
        z[x][y] = '$'; // 将走过的瓷砖做标记
        f(x-1, y);
        f(x+1, y);
        f(x, y-1);
        f(x, y+1);
 }
}
```

```
#include <iostream>
using namespace std;
int W, H;
char z[21][21];
int main()
{
 int i, j, num, sum;
 while(cin>>H && cin>>W && W != 0 && H != 0)
 {
        num = 0;
        sum = 0;
        for(i = 0; i < W; i++) // 读入矩阵
                cin>>z[i];
        for(i = 0; i < W; i++)
                for(j = 0; j < H; j++)
                        if(z[i][j] == '@') f (i,j);
        for(i = 0; i < W; i++)
                for(j = 0; j < H; j++)
                {
                        if(z[i][j] == '$') sum++;
                }
        cout<<sum<<endl;

 }
 return 0;
}
```

思路二

- “直接法”
  - 求的是从某点能到达的瓷砖数；
  - 假设有个函数能直接返回这个数；
  - 那么这个函数该如何写呢？
    （1）这个函数的原型应该长成什么样？
    （2）求得结果的计算过程应该如何描述？

```
11 6
..#..#..#..
..#..#..#..
..#..#..###
..#..#..#@.
..#..#..#..
..#..#..#..
```

$$f(x, y) = 1 + f(x - 1, y) + f(x + 1, y) + f(x, y - 1) + f(x, y + 1)$$

```
int f(int x, int y)
{
 if(x < 0 || x >= W || y < 0 || y >= H)
                                        // 如果走出矩阵范围
        return 0;
 if(z[x][y] == '#')
        return 0;
 else
 {
        z[x][y] = '#';                  // 将走过的瓷砖做标记
        return 1 + f(x - 1, y) + f(x + 1, y) + f(x, y - 1) + f(x, y +
        1);
 }
}
```

```
#include <iostream>
using namespace std;
int W, H;
char z[21][21];
int main()
{
 int i, j, num;
 while(cin>>H && cin>>W && W != 0 && H != 0)
 {
        num = 0;
        for(i = 0; i < W; i++) // 读入矩阵
              cin>>z[i];
        for(i = 0; i < W; i++)
              for(j = 0; j < H; j++)
                    if(z[i][j] == '@') cout<<f (i , j)<<endl;
 }
 return 0;
}
```

2007年计算概论A期末上机考试

- Problem ID Title
  - 3248 Problem A 最大公约数
  - 3249 Problem B 进制转换
  - 3246 Problem C 展览会
  - 3247 Problem D 回文素数

2008年计算概论A期末上机考试

- Problem ID Title
  - Problem A: 距离排序
  - Problem B: 寻找平面上的极大点
  - Problem C: 扩号匹配问题

2009年计算概论A期末上机考试

- Problem ID Title
  - 3708：1的个数
  - 2952：循环数
  - 1321：棋盘问题

议题5：从结构化到面向对象

——从 C 到 C++

北京大学

程序设计语言的发展

现实世界的问题

现实世界中的解决方案

运行的程序

程序设计语言的发展

现实世界的问题

现实世界的解决方案

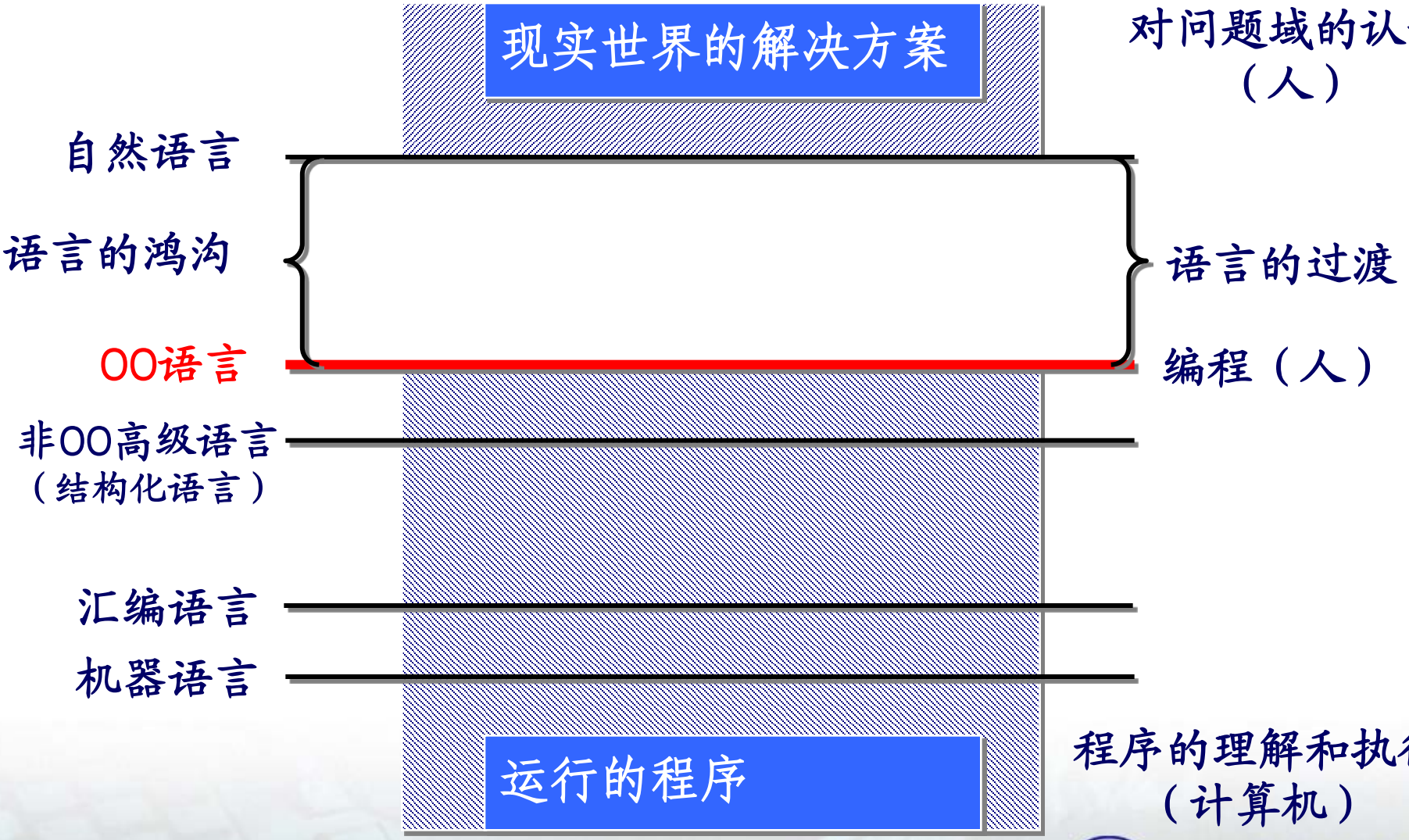
自然语言

语言的鸿沟

编程语言

运行的程序

程序设计语言的发展



北京大学

面向对象程序设计语言

- 基本原则
  - 万物皆为对象;
    - 对象表达为自身信息和可执行方法的组合;
  - 程序是对象的集合;
    - 对象直接可以通过发送消息调用对方;
  - 类似的对象可以抽象为类;
    - 具有相同类型的自身信息和可执行方法的对象可以抽象为类;
  - 类之间可以继承、聚合……;

```
#include<iostream>
#include<string>
using namespace std;
class Person{
private:
    string name;
public:
    Person(string input_name){
        name = input_name;
    }
    void showname(){
        cout<<name<<endl;
    }
};
```

```
class Student: public Person{
private:
    int id;
public:
    Student(int input_id, string input_name):Person(input_name){
        id = input_id;
    }
    void showid(){
        cout<<id<<endl;
    }
};
int main()
{
    Student mike(20, "Mike");
    mike.showid();
    mike.showname();
    return 0;
}
```

最后一个想讲而没能讲到的议题：

软件是怎样炼成的?

北京大学

我们接触到的软件

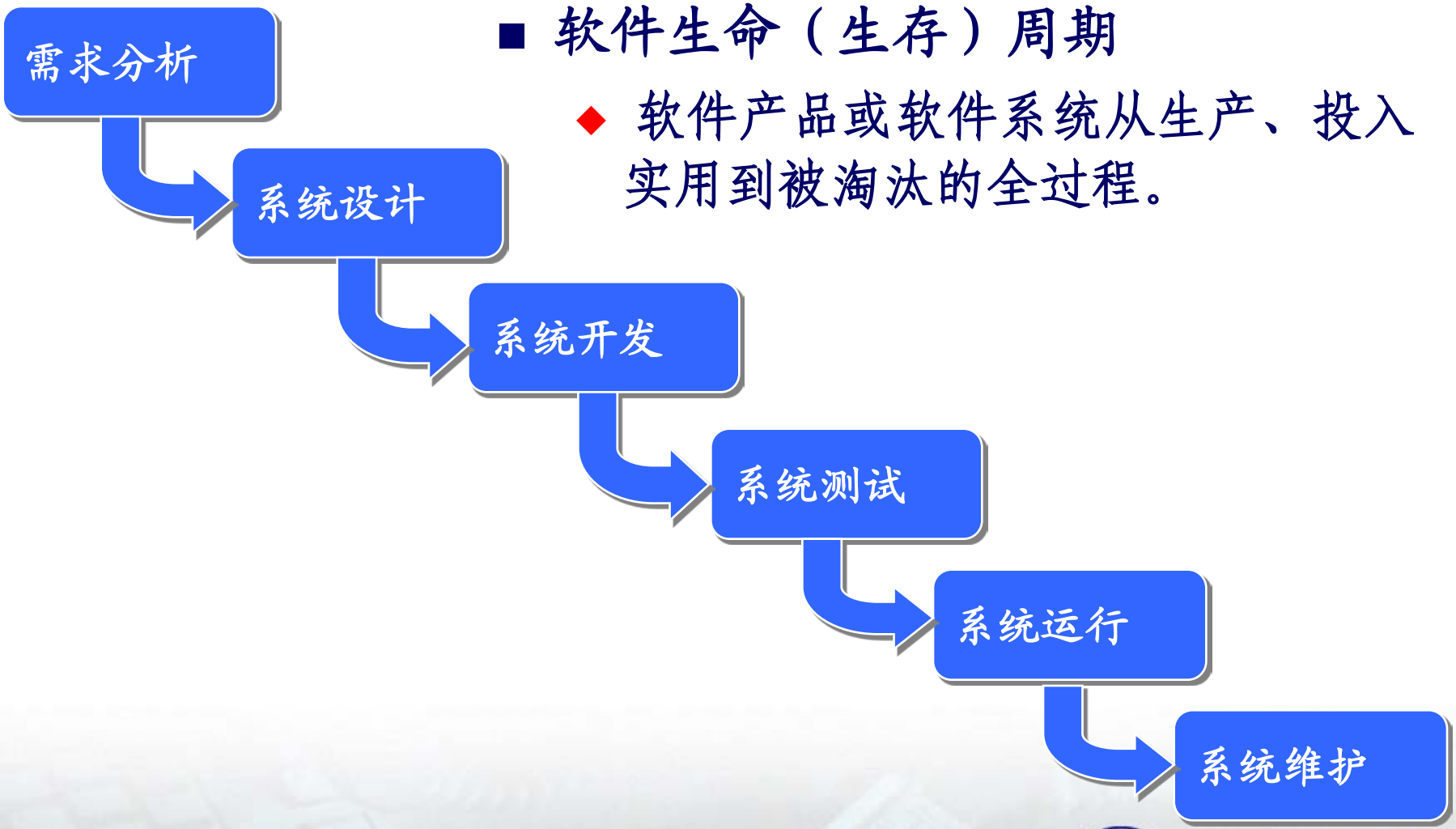
应用软件
其他系统软件
操作系统
硬件系统
数据库
Word
Linux
Unix
数学
软件包
game
图形包
Windows
程序设计
语言
Paintbrush
设备驱动程序
Calculator
Notepad

软件是如何开发出来的?

- 对软件的一种认识
  - 软件是对现实世界中解决方案的模拟，是现实世界的解决方案在计算机系统中的映射;
  - 软件的开发过程是建立现实世界的解决方案在计算机系统中的映射的过程;
  - 从认识论的角度，软件开发从本质上讲就是对软件所要处理的问题域进行正确的认识，并把这种认识正确的描述出来。
    - 开发一个软件是为了解决某些问题，这些问题所设计的业务范围，称作该软件的问题域。

传统的软件生命周期

- 软件生命（生存）周期
  - 软件产品或软件系统从生产、投入实用到被淘汰的全过程。



北京大学

相关知识简介

- 需求分析
- 系统设计
- 系统开发
- 系统测试
- 系统运行
- 系统维护

- 主要问题：
  - 如何获取用户需求
  - 如何描述获取到的需求
- 相关的分析方法：
  - 功能分解法
  - 面向数据流的分析方法
  - 面向对象的分析方法
- 相关的描述方法：
  - 自然语言的描述方式
  - 形式化的描述方法

相关知识简介

- 需求分析
- 系统设计
- 系统开发
- 系统测试
- 系统运行
- 系统维护

- 主要问题:
  - 如何勾画系统的蓝图
  - 如何指导系统的开发
- 相关的方法与技术:
  - 软件建模技术
    - UML
  - 软件体系结构
  - 软件构件技术

相关知识简介

- 需求分析
- 系统设计
- 系统开发
- 系统测试
- 系统运行
- 系统维护

- 主要问题:
  - 如何把设计转换为程序
  - 如何管理开发的成果
- 相关的方法与技术:
  - 软件开发环境
    - Visual Studio, Eclipse
  - 程序设计语言
    - Basic, C, C++, C#, Java
  - 软件配置管理技术

相关知识简介

- 需求分析
- 系统设计
- 系统开发
- 系统测试
- 系统运行
- 系统维护

- 主要问题：
  - 如何帮助用户开发正确的程序
  - 如何验证程序的正确性等。
- 相关的方法与技术：
  - 软件调试技术
    - 如何定位错误
    - 如何辅助开发者修改错误
  - 软件测试技术
    - 黑盒测试，白盒测试
    - 单元测试，集成测试

相关知识简介

- 需求分析
- 系统设计
- 系统开发
- 系统测试
- 系统运行
- 系统维护

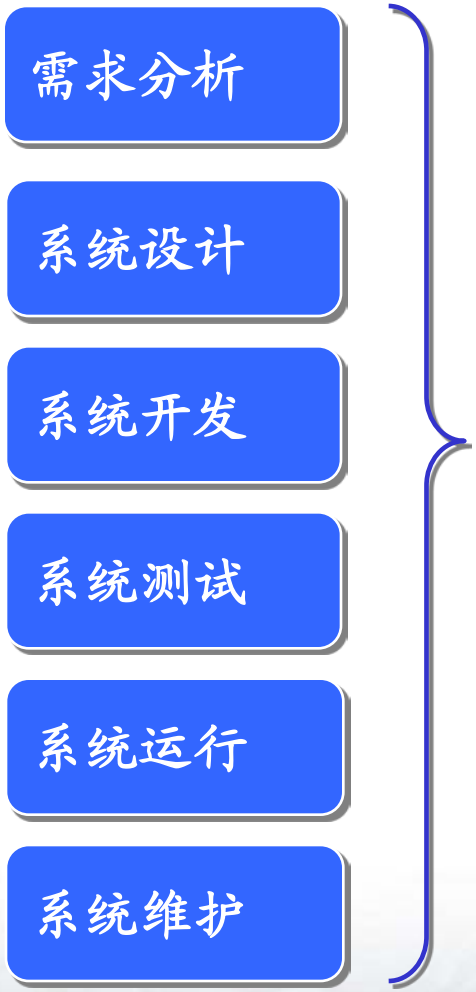
- 主要问题:
  - 如何提高程序运行的效率
  - 如何保证程序运行的安全性、可靠性、动态调整性。
- 相关的技术:
  - 操作系统的设计;
  - 软件中间件技术;
    - 应用服务器（Weblogic, Websphere等等）

相关知识简介

- 需求分析
- 系统设计
- 系统开发
- 系统测试
- 系统运行
- 系统维护

- 主要问题:
  - 如何应对需求的变化;
  - 如何抽取系统中可以复用的成分;
- 相关的技术:
  - 程序理解技术
    - 帮助人们理解程序
  - 软件逆向工程方法
    - 由成果获得规划

相关知识简介



- 主要问题
  - 如何通过管理、控制生产过程保证软件质量
- 相关知识点：
  - 软件过程模型
    - 敏捷软件过程
  - 软件过程度量
    - CMM

感谢各位同学！

希望能在软件研究所见到大家！

北京大学