计算机科学基础

——程序设计与算法语言

第二章 基本控制结构程序设计

戚隆宁

Email: longn_qi@seu.edu.cn

Tel: 13813839703

本章目录

算法的概念与表示方法

- 算法的概念
  - 用于解决特定问题的有限计算方法或执行序列
- 算法的特征
  - 至少一个输出，任意多输入
  - 有效性：计算结果正确
  - 有穷性：计算步骤有限
  - 确定性：输入相同结果相同

算法的概念与表示方法

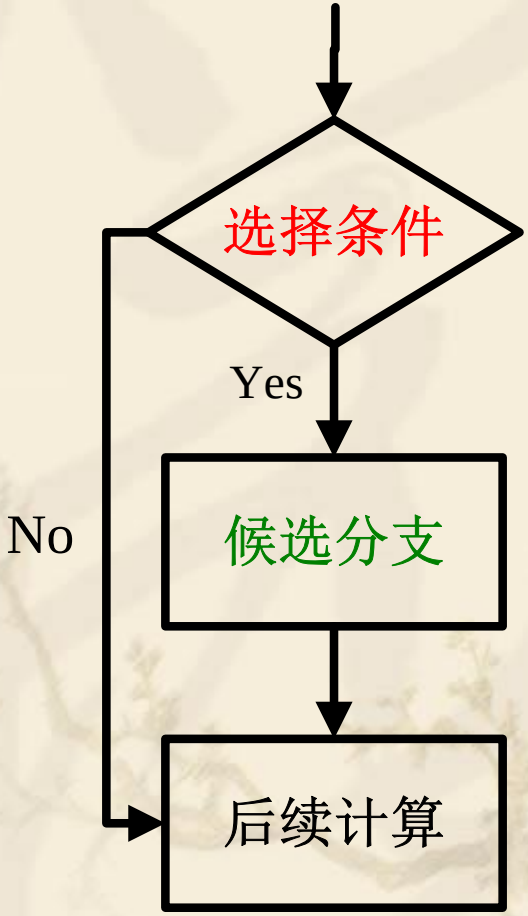
- 算法的逻辑控制结构
  - 顺序结构
  - 分支结构
  - 循环结构
- 算法的表示
  - 语言：自然语言、类计算机语言（伪代码，Pseudocode）、计算机语言（代码）
  - 图形：流程图、N-S图、PAD图
  - 表格

分支结构的程序设计

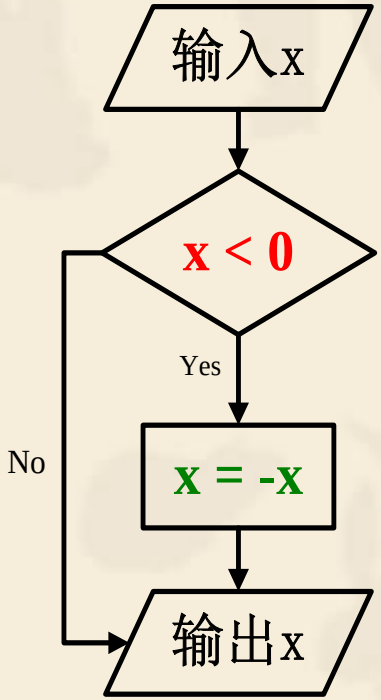
- 缘由：需要根据条件满足与否选择执行动作
- 分支结构
  - 基本要素
    - 若干候选分支
    - 选择条件
  - 由判断的结果决定选取一条分支执行
    - 单分支
    - 双分支
    - 多分支

分支结构的程序设计

- 单分支选择（开关选择）
  - 选择条件满足时执行分支，否则跳过



分支结构的程序设计



伪代码：

输入x
if x < 0
then x ← -x
输出x

C++语句：

```
int x;
cin >> x;
if ( x < 0 ) x = -x;
cout << x;
```

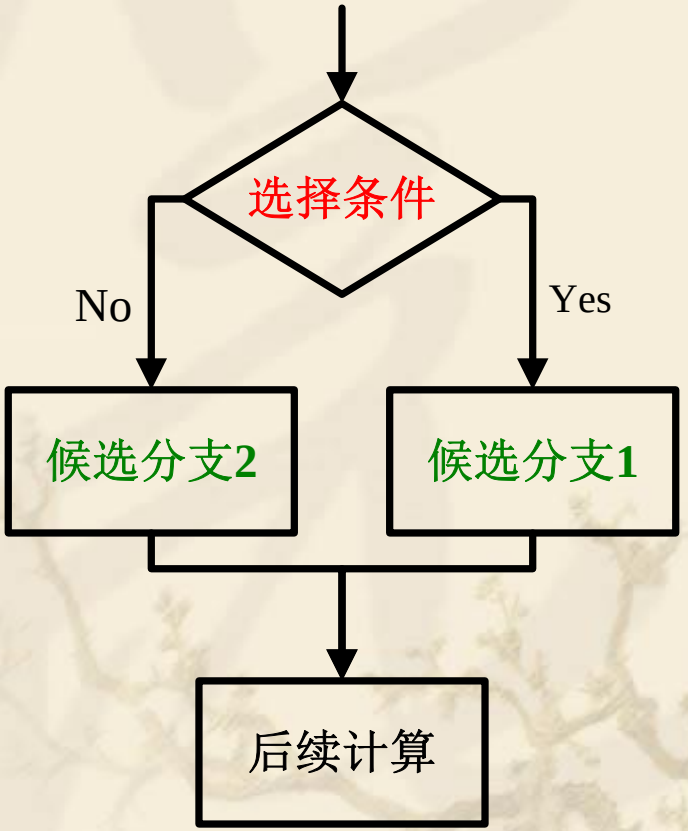
判断条件表达式的值必须是**bool**型或可转换为**bool**型

注意：

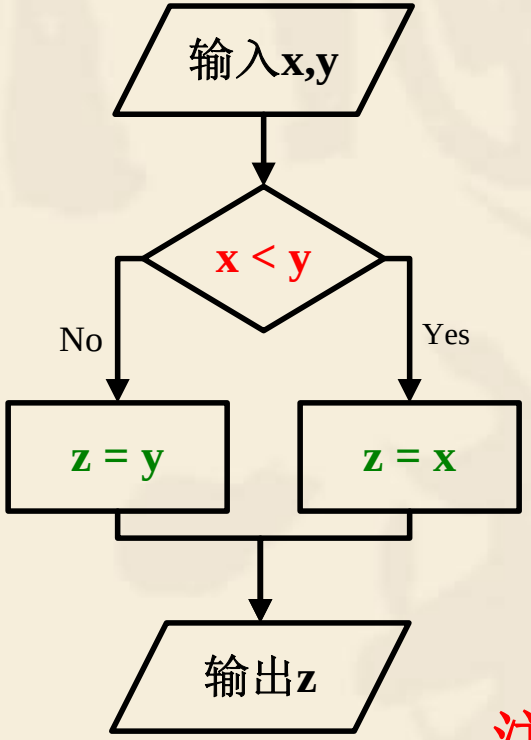
if分支只能是单语句或复合语句（语句块）

分支结构的程序设计

- 双分支选择（互斥选择）
  - 两个互斥候选分支
  - 根据条件判断结果选择一个分支执行



分支结构的程序设计



伪代码：

输入x和y

if x < y

then z ← x

else z ← y

输出z

C++语句：

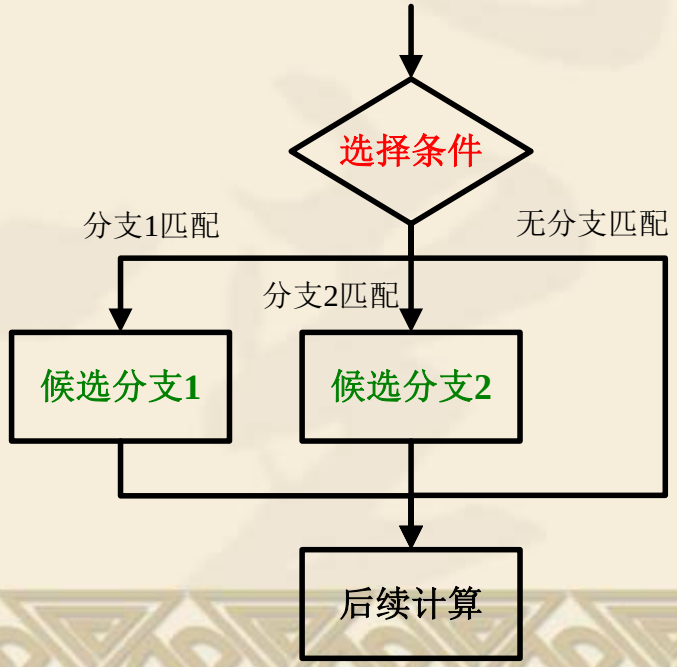
```
int x, y, z;
cin >> x >> y;
if ( x < y )
{ z = x; }
else { z = y; }
cout << z;
```

注意：

else分支只能是单语句或复合语句（语句块）

分支结构的程序设计

- 多分支选择
  - 有多个候选分支
  - 若无匹配分支则跳过



C++语句：

```
int y = 0;
char x;
cin >> x;
switch ( x )
{
    case '+': y += 10; break;
    case '-': y -= 10; break;
    default: y = -1;
}
cout << y;
```

判断条件表达式的值必须是整型或可提升为整型

分支条件的值必须是整型或可提升为整型的常量

分支结构的程序设计

- 条件运算符（?:）
  - 逻辑条件？表达式1：表达式2
  - 条件成立结果为表达式1，否则结果为表达式2
- 分支嵌套
  - 允许候选分支中出现分支结构
  - else与if的配对：就近未配对原则
- 分支判断的优化
  - 当条件为多个表达式的“与逻辑”（&&）组合时，只要有表达式为false，后续表达式将不再计算。
  - 当条件为多个表达式的“或逻辑”（||）组合时，只要有表达式为true，后续表达式将不再计算。

分支结构的程序设计

- 分支嵌套形式1

```
if (…)
{
    if (…) { }
    else { }
}
else
{
    if (…) { }
    else { }
}
```

- 分支嵌套形式2

```
if (…)
{
    if (…) { }
    else { }
}
else if (…)
{ }
else
{ }
```

- 分支嵌套形式3

```
if (…)
    if (…) { }
    else { }
else
{
    if (…) { }
    else { }
}
```

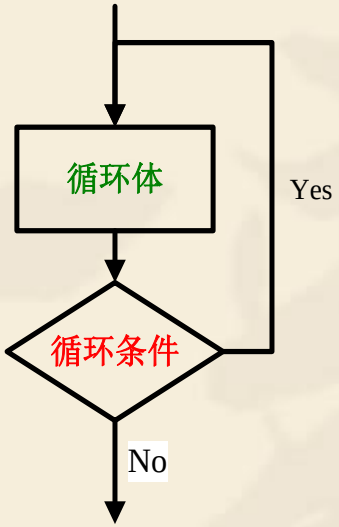
注意：else分支与if就近配对
建议：用{}区分清楚

循环结构的程序设计

- 缘由：具有重复性的计算或操作
- 循环结构的基本要素
  - 循环体
  - 循环条件
- 循环结构
  - 直到型循环（后置循环条件）
  - 当型循环（前置循环条件）
    - 计数型循环

循环结构的程序设计

- 直到型循环（until循环）
  - 后置循环条件：执行直到循环条件不满足为止
  - 至少执行循环体1次



伪代码：

```
x ← 0
repeat x ← x + 1
until x ≥ 100
输出 x
```

C++语句：

```
int x = 0;
do { x = x + 1; }
while ( x < 100 );
cout << x;
```

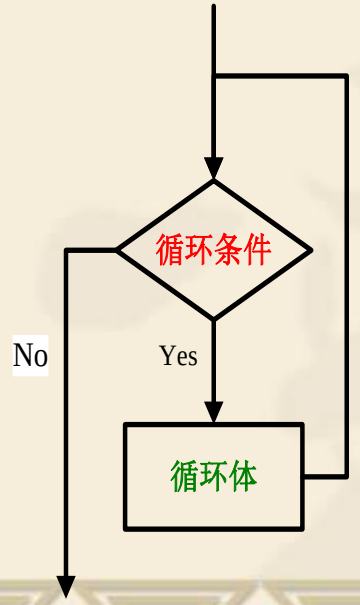
判断条件同if
循环体可以为空

累加输入的100个数

```
int x; // 保存输入的数据
int y = 0; // 保存累加结果
int i = 0; // 累加次数
do
{
        cin >> x;      // 输入数据
        y += x;        // 累加数据
}while ( ++i < 100 ); // 累加计数和判断
cout << y << endl; // 输出累加结果
```

循环结构的程序设计

- 当型循环（while循环）
  - 前置循环条件：条件满足才执行循环
  - 循环体有可能不被执行



伪代码：

```
x ← 0
while x < 100
do x ← x + 1
输出x
```

C++语句：

```
int x = 0;
while ( x < 100 )
 { x = x + 1; }
cout << x;
```

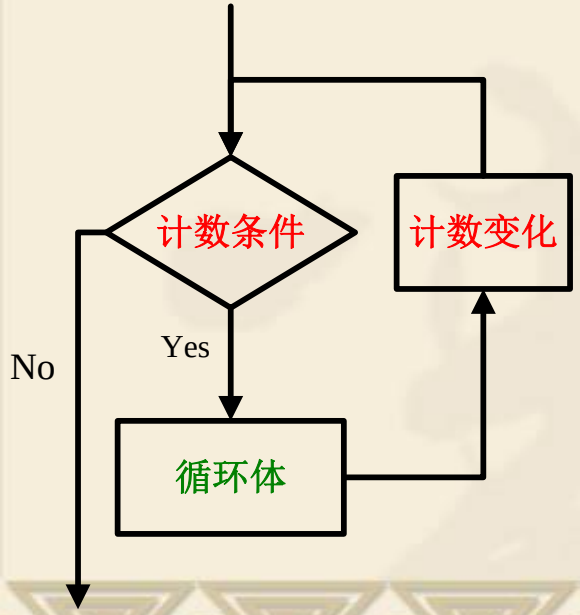
判断条件同if
循环体可以为空

累加输入的100个数

```
int x; // 保存输入的数据
int y = 0; // 保存累加结果
int i = 0; // 累加次数
while ( i++ < 100 ) // 累加计数并判断
{
        cin >> x;     // 输入数据
        y += x;       // 累加数据
}
cout << y << endl; // 输出累加结果
```

循环结构的程序设计

- 计数型循环（for循环）
  - 以计数值的判断作为循环前置条件
  - 需要有初始计数值和计数变化（递增或递减）



伪代码：

x ← 0

for i ← 1 to 10

do x ← x + i

输出x

C++语句：

```
int x = 0, i;
for ( i=1; i<=10; i++ )
 { x = x + i; }
cout << x;
```

初始条件、计数条件、计数累加和循环体都可以为空

累加输入的100个数

```
int x; // 保存输入的数据
int y = 0; // 保存累加结果
int i = 0; // 累加次数
for ( ;i < 100; i++ ) // 累加计数并判断
{
        cin >> x;     // 输入数据
        y += x;       // 累加数据
}
cout << y << endl; // 输出累加结果
```

循环结构的程序设计

- 循环条件判断的优化
  - 同分支条件判断的优化
- 循环嵌套
  - 允许循环体中使用循环结构（多重循环）

```
for (…)
{
    for (…) { }
}
```

```
for (…)
    for (…)
    {
    }
```

```
while (…)
{
    for (…) { }
}
```

显示5行6列的星号矩形

```
******
******
******
******
******

******
*     *
*     *
*     *
******
```

```
int i, j;
for( i = 1; i <= 5; i++ )
{
        for( j = 1; j <= 6; j++ )
                cout << '*';
        cout<<endl;
}
```

空心矩形框呢？

循环结构的程序设计

- 循环的改变
  - continue：跳过后续语句继续循环
  - break：结束当前循环（不能结束嵌套循环）
  - goto：可跳出多重循环

统计一个整数二进制表示中1的个数

```
int x, count = 0, i;
cin >> x;
for( i = 0; i < 32; i++ )
{
        if( ( x & ( 1 << i ) ) == 0 )
                continue;
        count++;
}
cout << count << endl;
```

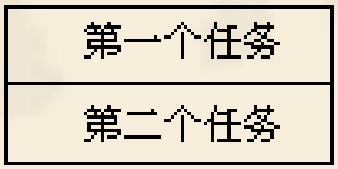
continue：跳过后续语句继续循环

判断一个整数是否是素数

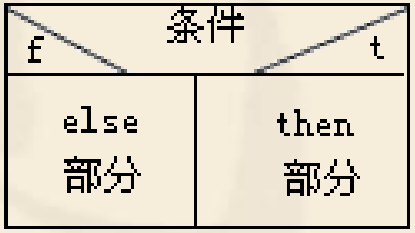
```
int x, i;
cin >> x;
for( i = 2; i < x; i++ )
{
        if( x % i == 0 )
                break;
}
if ( i == x )
        cout << x << “是素数” << endl;
else
        cout << x << “不是素数” << endl;
```

break：结束当前循环（不能结束嵌套循环）

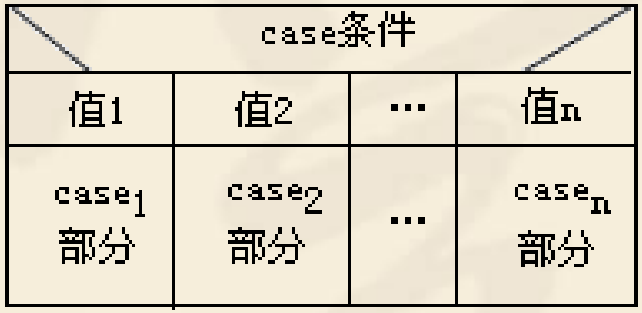
N-S图表示



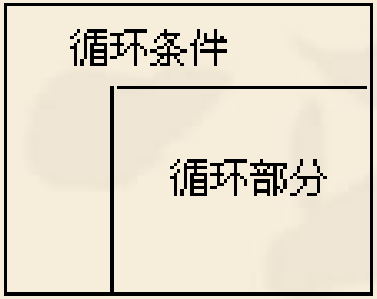
(a)顺序



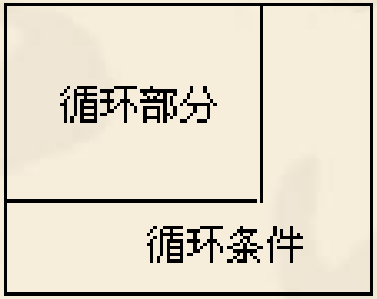
(b)分支



(c)case



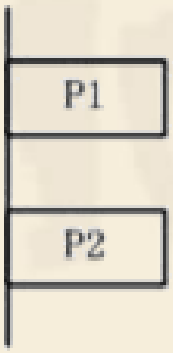
(d)do-while循环



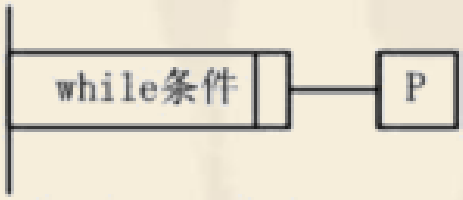
(e)do-until循环

(f)调用子程序

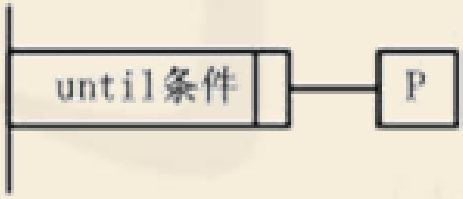
PAD图表示



(a)顺序

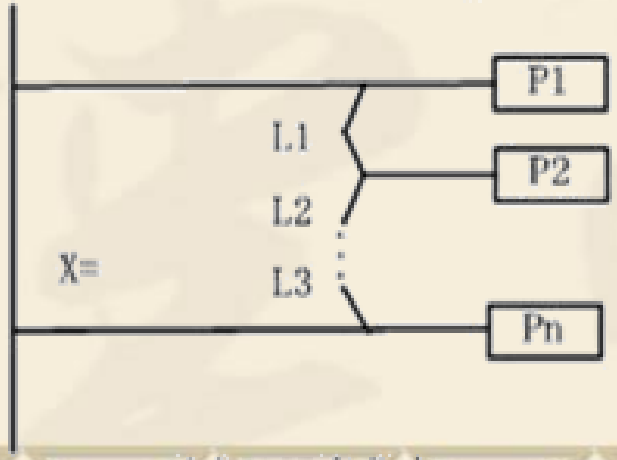


(b)while 循环



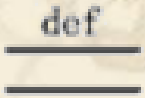
(d)until 循环

(c)if 选择



(e)CASE多分支

(f)语句标号



(g)定义

结构化程序设计思想

- 面向过程
  - 程序 = 算法 + 数据结构
  - 数据与数据处理分离
  - 自顶向下，逐步细化
- 缺点
  - 不适合大型软件
  - 可重用性差

常用算法应用实例

- 穷举法
- 筛选法
- 递推法

常用算法应用实例1

- 穷举法
  - 前提：问题的候选解有限且范围已知，候选解的验证方法简单
  - 基本思路：逐个验证候选解的有效性
- 例子
  - 求100以内的所有素数
  - 百鸡问题（不定方程求解问题）
    - 公鸡一只值钱五，母鸡一只值钱三，小鸡三只值钱一，百钱买百鸡，问可买公鸡、母鸡和小鸡各多少只？
  - 密码破译（排列组合问题）

求100以内素数及个数

```
int x, i, n=0;
for( x = 1; x < 100; x++ )
{
        for( i = 2; i < x; i++ )
                if( x % i == 0 ) break;
        if ( i == x )
                cout << “找到第”<<++n<<“个素数:”
                        << x << endl;
}
if ( n == 0 )
        cout << “找不到素数！ ” << endl;
```

百鸡问题

设公鸡x只，母鸡y只，小鸡z只

$$\begin{cases} x+y+z=100 \\ 5x+3y+\dfrac{z}{3}=100 \\ x,y,z \in N \end{cases}$$

```
int cocks,hens,chickens;
for( cocks = 0; cocks <= 20; cocks++ )
    for( hens = 0; hes <= 33; hes++ )
    {
        chickens = 100-cocks-hens;
        if(5*cocks + 3*hens + chickens/3 == 100 )
            cout << “公鸡” << cocks << “只，”
                 “母鸡” << hens <<“只，”
                 “小鸡” << chickens << “只” << endl;
    }
```

百鸡问题

设公鸡x只，母鸡y只，小鸡z只

$$\begin{cases} x+y+z=100 \\ 5x+3y+\dfrac{z}{3}=100 \\ x,y,z \in N \end{cases}$$

```
int cocks,hens,chickens;
for( cocks = 0; cocks <= 20; cocks++ )
    for( hens = 0; hes <= 33; hes++ )
    {
        chickens = 100-cocks-hens;
        if( chickens%3 == 0 && 5*cocks + 3*hens + chickens/3
== 100 )
            cout << “公鸡” << cocks << “只，”
                 “母鸡” << hens <<“只，”
                 “小鸡” << chickens << “只” << endl;
    }
```

常用算法应用实例2

- 筛选法/排除法
  - 前提：问题的候选解有限且范围已知，候选解的排除方法简单且可以穷尽
  - 基本思路：逐个排除候选解
- 例子
  - 求100以内的所有素数
  - 数独问题

常用算法应用实例3

- 递推法/迭代法
  - 前提：问题的解可以从一个初始值通过有限次重复递推得到
  - 基本思路：确定递推关系，根据初始条件或最终状况进行递推
  - 基本方法
    - 顺推法
    - 逆推法
- 例子
  - 阶乘n!，幂级数展开 $e^x = \lim_{n\to\infty} \sum_{i=0}^{n} \frac{x^i}{i!}$
  - 兔子繁殖问题（Fibonacci数列）
    - 兔子在出生两个月后，就有繁殖能力，一对兔子每个月能生出一对小兔子来。如果所有兔都不死，那么一年以后可以繁殖多少对兔子？
  - 杨辉三角（二项式系数）

$n! = 1 \times 2 \times 3 \times \ldots \times n$

$0! = 1$

$f(n)=f(n-1) \times n$

$f(0)=1$

阶乘计算

```
int n, y;
cin >> n;
y = 1; // f(0)=1
for( i = 1; i < n; i++ )
{
        next_y = y * i; // f(i)=f(i-1) ×i
        y = next_y;
}
cout << y << endl;
```

$n! = 1 \times 2 \times 3 \times \ldots \times n$

$0! = 1$

$f(n)=f(n-1) \times n$

$f(0)=1$

阶乘计算

```
int n, y;
cin >> n;
y = 1; // f(0)=1
for( i = 1; i < n; i++ )
{
        y *= i; // f(i)=f(i-1) × i
}
cout << y << endl;
```

$f(n)=\sum_{i=0}^{n}\frac{x^i}{i!}, f(0)=1$

$f(n)=f(n-1)+\frac{x^n}{n!}$

幂级数展开求e^x

$g(n)=\frac{x^n}{n!}, g(0)=1$

$g(n)=g(n-1)\frac{x}{n}$

```
int i;
double x, y = 1, delta_y = 1;
cin >> x; // e的指数
for( i = 1; delta_y > 5e-7; i++ )
{
        delta_y *= x/i; // g(n) = g(n-1) x/n
        y += delta_y; // f(n) = f(n-1) + g(n)
}
cout << "y=" << y << endl;
cout << "迭代" << i << "次" << endl;
```

$\{1, 1, 2, 3, 5, 8, 13 \ldots\}$

$f(n) = f(n-1) + f(n-2)$

$f(0) = 0, f(1) = 1$

斐波那契数列

```
int y, y_1 = 1, y_2 = 0;
cin >> n;
for( i = 2; i <= n; i++ )
{
        y = y_1 + y_2;
        y_2 = y_1;
        y_1 = y;
}
cout << "f("<<n<<")=" << y << endl;
```

$\{1, 1, 2, 3, 5, 8, 13 \ldots\}$

$f(n) = f(n-1) + f(n-2)$

$f(0) = 0, f(1) = 1$

斐波那契数列

```
int y = 1, y_1 = 1, y_2 = 0;
cin >> n;
for( i = 2; i <= n; i++ )
{
        y = y_1 + y_2;
        y_2 = y_1;
        y_1 = y;
}
if ( n >= 1 )    cout << "f("<<n<<")=" << y << endl;
else cout << “n应为正整数" << endl;
```

枚举类型

- 枚举常量：用标识符表示的整型常量（const int）
- 枚举类型：枚举常量的集合
- 枚举类型的定义语法

enum 枚举类型名 { 枚举常量表 };

例：enum WEEK { Sun, Mon, Tue, Wed, Thu, Fri, Sat };

- 枚举常量的值默认从0开始，默认递增1，允许同值
- 枚举类型不能有交集
- 枚举变量的定义

enum 枚举类型名 变量名1，变量名2;

例：WEEK a = Mon, b;

枚举类型的运算

- 枚举类型变量的运算
  - 提升到整型参与运算
  - 只接受同类型枚举变量或常量赋值（不支持复合赋值、自增、自减）
  - 不能直接输入，可以输出（输出枚举常量值）