

第三章 结构化程序设计

——0301 输入输出与顺序结构

丁子君

0301 输入输出与顺序结构

- 3.1 算法的基本概念
- 3.2 C/C++程序的基本结构、语句的种类
- 3.3 C++的输入和输出
- 3.4 C++的格式化输入和输出
- 3.5 顺序结构程序的编写

3.1 算法的基本概念

1. 定义：在程序设计上表现为一组**指令序列**，给出解决问题的逻辑描述，根据算法描述进行实际编程；

2. 算法：数值算法+非数值算法

3. 程序 = 算法 + 数据结构

算法：对操作的描述，计算机处理问题的步骤；

数据结构：对数据的描述，计算机用什么方式存储相关的数据信息；

4. 表达方式：自然语言、流程图、伪代码、计算机语言

3.1 算法的基本概念

➤例：设计算法：输入圆的半径，输出圆的面积

1. 自然语言：

第一步：定义 $\pi=3.14$

第二步：输入圆的半径

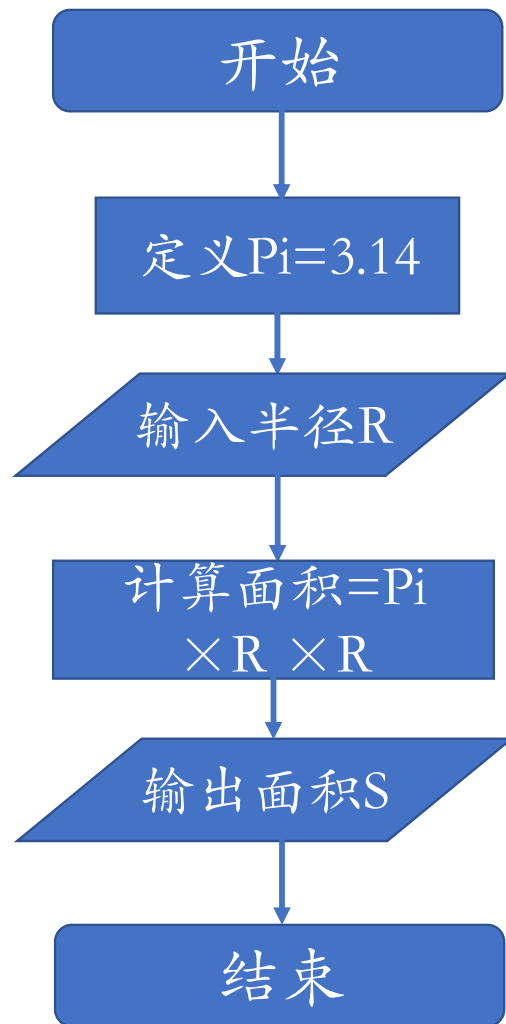
第三步：用 $s=\pi r^2$ 计算面积




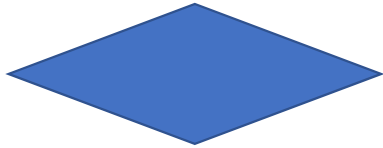
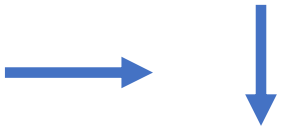
第四步：输出圆的面积

文字冗长，很容易造成歧义（二义性）

3.1 算法的基本概念

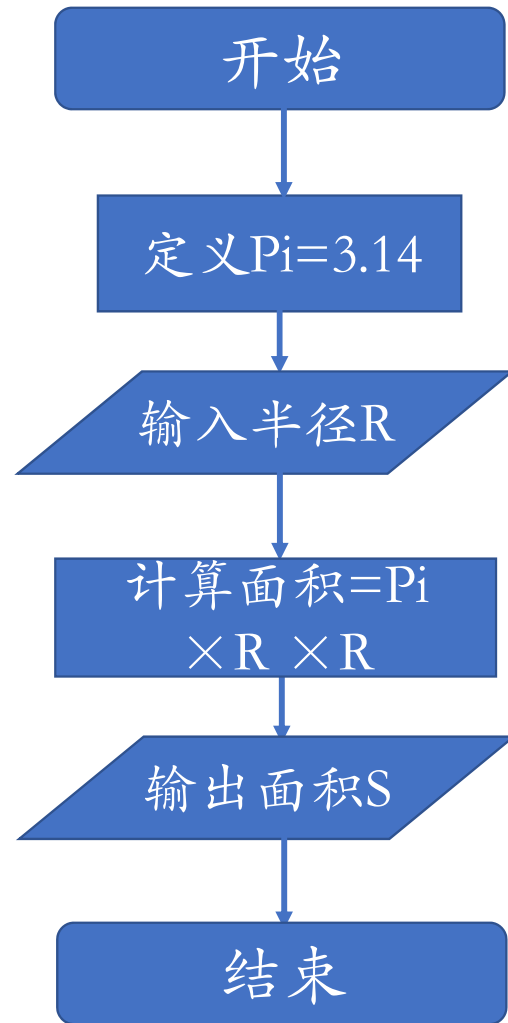
2. 流程图：



	圆角矩形，开始和结束符号
	矩形，流程符号，表示操作步骤
	平行四边形，表示输入和输出
	菱形，判断符号，表示算法中的判定条件（分支点）
	箭头，流程线符号，表示算法中的流程进展方向

3.1 算法的基本概念

2. 流程图：



篇幅较大，对于复杂的程序，
难以阅读和修改

3.1 算法的基本概念

3. 伪代码：使用伪代码描述从1开始的连续n个自然数求和的算法

1. 算法开始
2. 输入n的值

3. $i \leftarrow 1$ // 为变量i赋初始值

4. $\text{sum} \leftarrow 0$ // 为变量sum赋初始值

5. while $i \leq n$ // 当变量 $i \leq n$ 时，执行下面的循环体语句

6. {

7. $\text{sum} \leftarrow \text{sum} + i$

8. $i \leftarrow i + 1$

9. } // 复合语句

10. 输出sum的值

11. 算法结束

采用自然语言和计算机高级语言的控制结构来描述操作步骤

3.1 算法的基本概念

5. 特征:

- (1) 输入
- (2) 输出
- (3) 有穷性
- (4) 确定性
- (5) 可行性

3.1 算法的基本概念

(1) 输入

一个算法有0个输入或者多个输入

(2) 输出

一个算法要有1个或者多个输出

(3) 有穷性:

必须在有限次执行后完成

- 1、步骤有限
- 2、每个步骤的时间有限（而且要合理）

例:

算法1

第一步: 令 n 等于0

第二步: n 加1

第三步: 转向第二步



算法2

第一步: 令 n 等于0

第二步: n 加1

第三步: 如果 n 小于10则转向第二步, 否则终止



3.1 算法的基本概念

(4) **确定性**: 算法中的每一步都必须有明确的定义, 不能有语义不明确的地方, 使产生歧义, 相同的输入不能结果不同

(5) **可行性**: 执行的任何计算步骤都是可执行的操作步骤 (按照当前算法可以得出结果, 是可行的)

例如:

求和: $S=1+2+3+\dots+\infty$ // **✗ 计算机不能实现和表示无穷大**

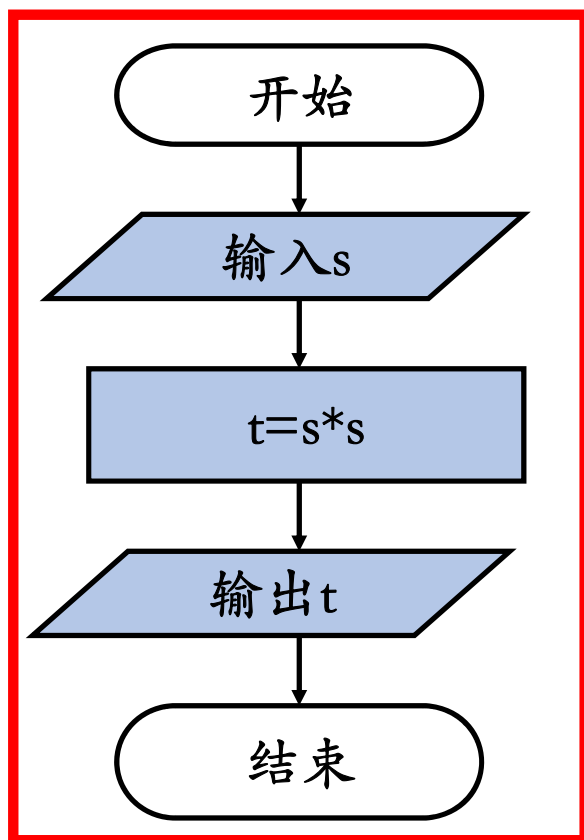
3.2 C/C++程序的基本结构和语句种类

- 程序：由计算机语言组成的语句序列

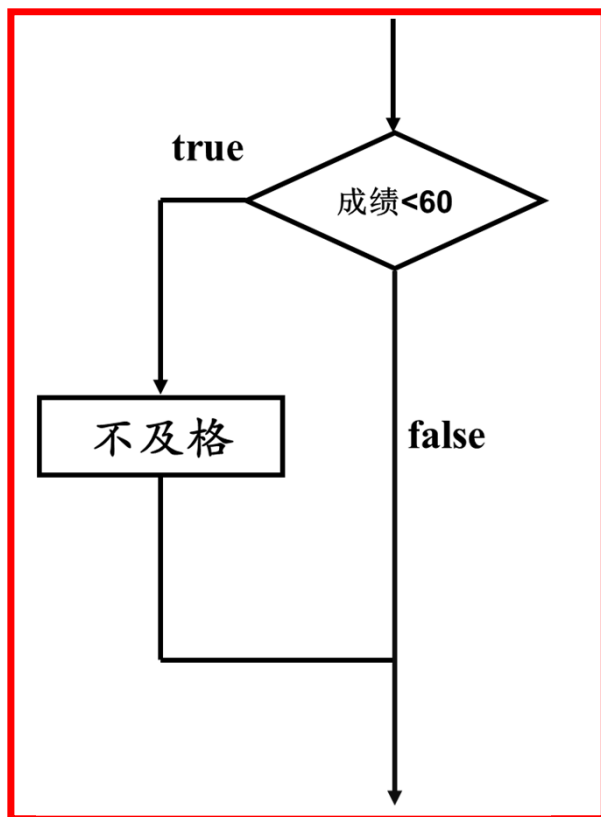


1. 顺序结构程序：最简单，最基本的结构，程序运行时，按语句的书写次序依次进行

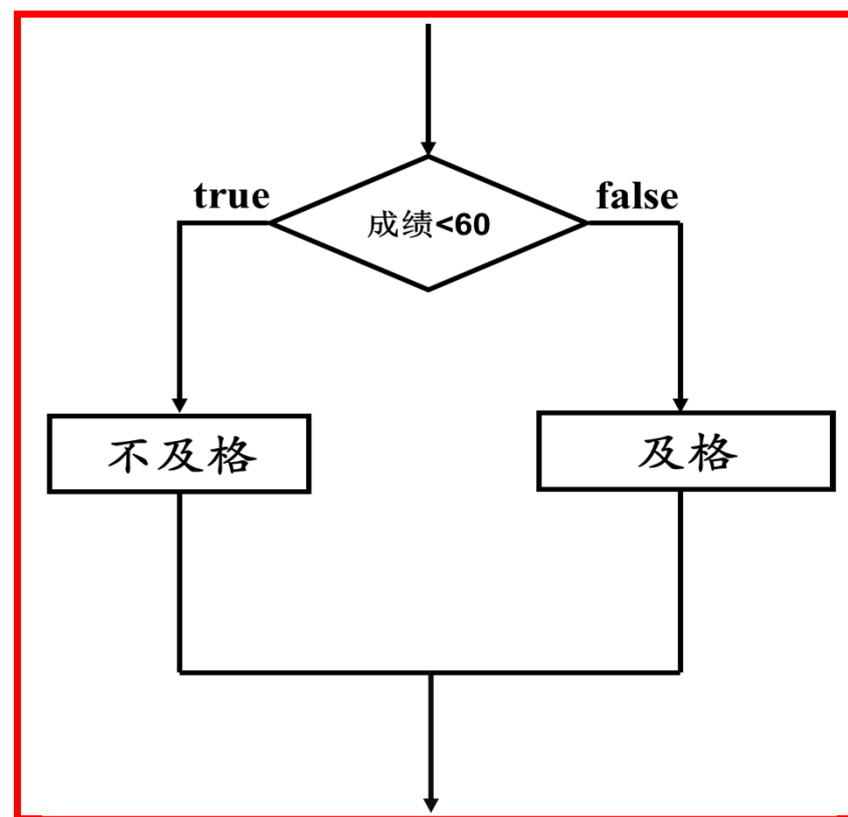
例：输出一个数字的平方



2. 选择(分支)结构程序：根据给定的条件进行判断，由判断结果来确定执行不同的语句

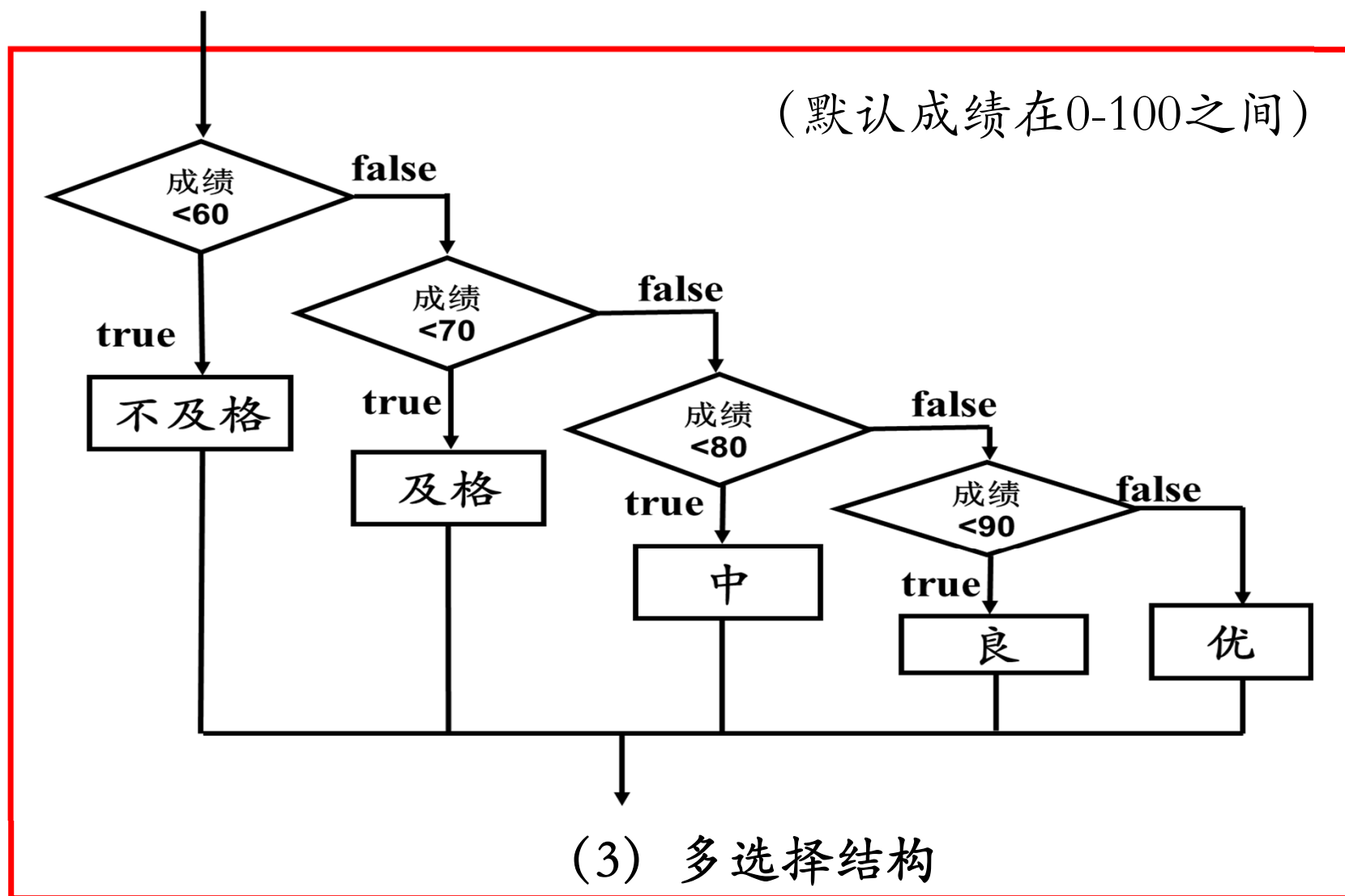


(1) 单选择结构



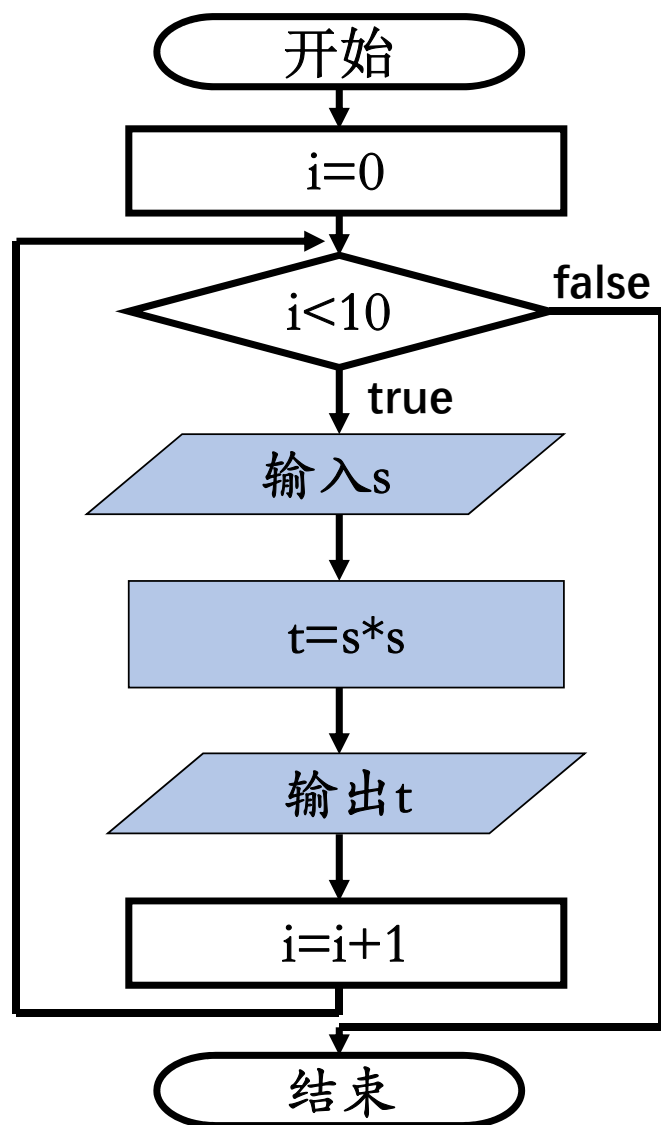
(2) 双选择结构

2. 选择(分支)结构程序:

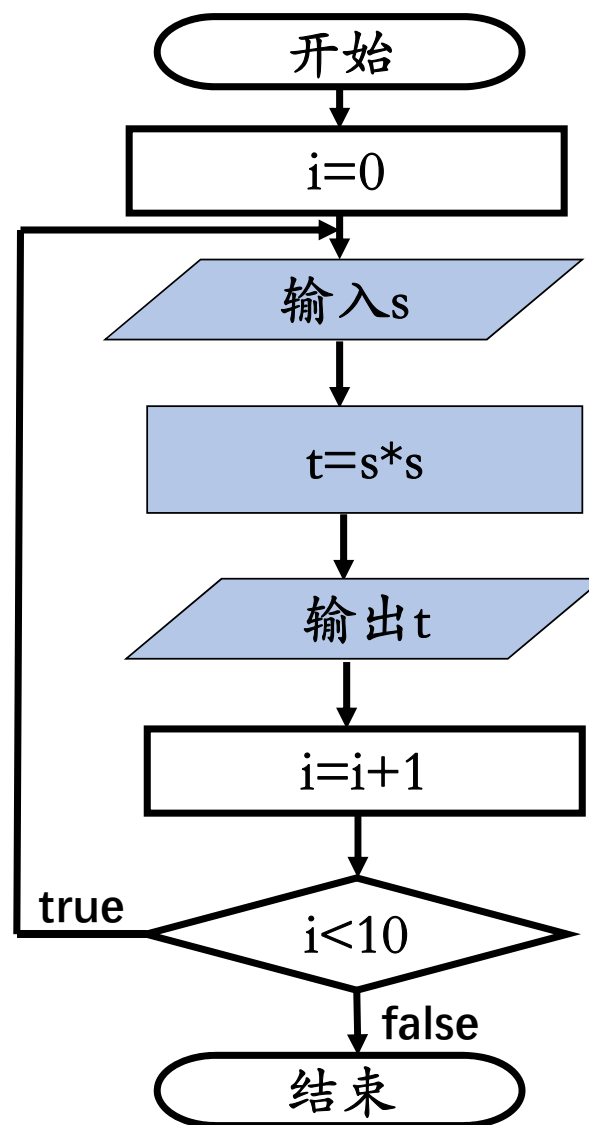


3. 循环结构程序：根据给定的条件进行判断，当条件成立时，重复执行某个或某些操作

(1) 当型循环

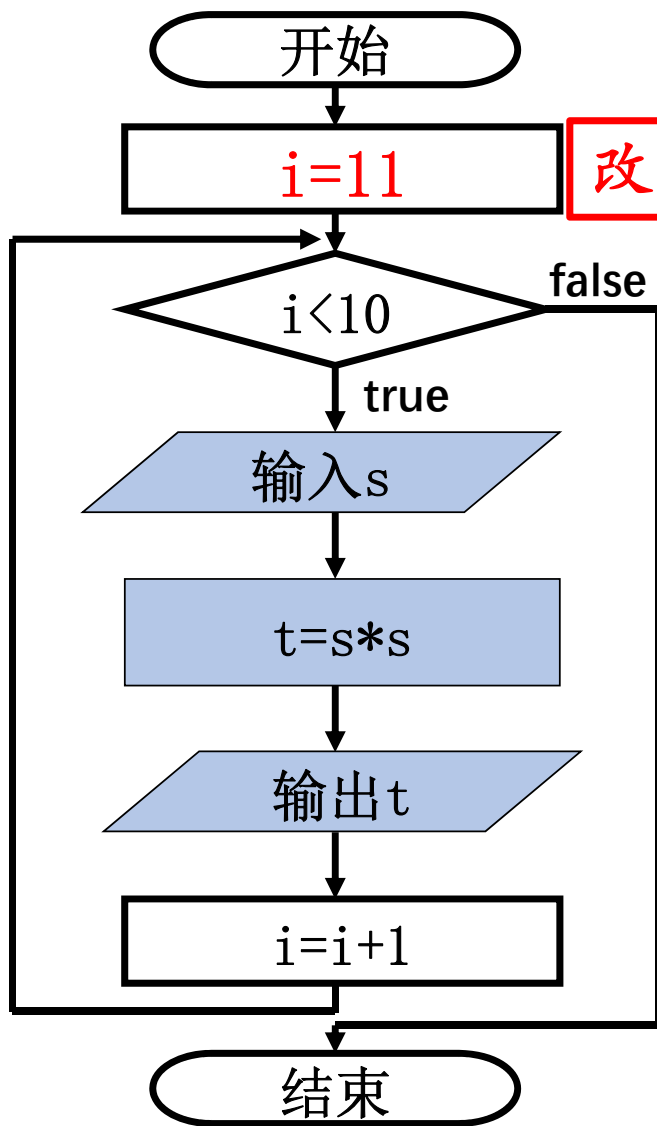


(2) 直到型循环



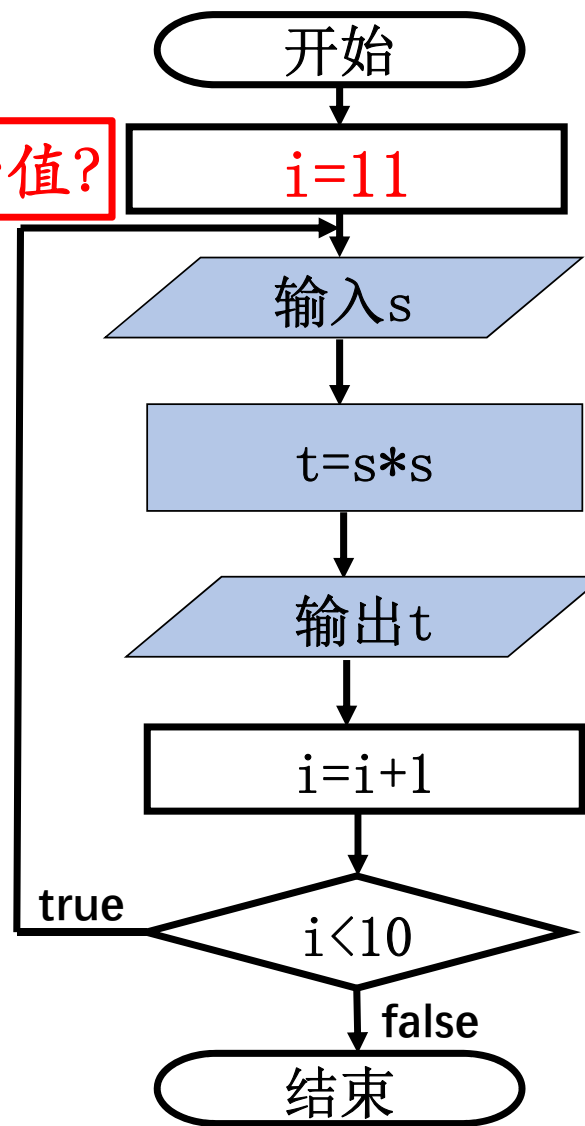
3. 循环结构程序：根据给定的条件进行判断，当条件成立时，重复执行某个或某些操作

(1) 当型循环



改变初始值?

(2) 直到型循环



3.2 C/C++程序的基本结构和语句种类

- 定义:

C++程序中最小的独立单位是语句，语句是指要执行的操作。语句由分号“;”结束

- 语句种类:

1. 定义语句
2. 赋值语句
3. 执行语句
 - (1) 控制语句
 - (2) 函数和流对象调用语句
4. 空语句
5. 复合语句

3.2 C/C++程序的基本结构和语句种类

1. 定义语句：指出要存储的数据类型和存储在这里的数据名称

例：int i;

int j;

➤ 定义变量i和j必须是整型，只声明，没有实际操作执行（非执行语句）
通常在首次使用前定义该变量

2. 赋值语句：将值赋给变量，由赋值表达式加上分号构成（回顾第二章）

例：1、为什么 int a=b=3; 不正确

2、为什么 int a, b;

a=b=3; 正确

3.2 C/C++程序的基本结构和语句种类

3. 执行语句

(1) 控制语句：完成一定的流程控制功能

①if() ~ else (条件语句)

②switch (多分支选择语句)

➤ 选择结构控制语句

③for() ~ (循环语句)

④while() ~ (循环语句)

⑤do ~ while (循环语句)

➤ 循环结构控制语句

⑥continue (结束本次循环语句)

⑦break (终止执行switch或循环语句)

⑧goto (转向语句)

➤ 转移结构控制语句

⑨return (从函数返回语句)

(2) 函数和流对象调用语句：由函数调用语句和分号构成

例：sort(x, y, z);

cout << x << endl;

3.2 C/C++程序的基本结构和语句种类

4. 空语句：语句以分号作为结束标志，因此单一的分号“；”也算一条语句，它是条空语句，作用是什么也不执行。在程序中，空语句主要用来做空循环体(扩充新功能)

```
#include <iostream>
using namespace std;

int main()
{
    int i, s ;
    for (i = 1, s = 0; i < 5; s += i++)
        cout << "s=" << s << endl;
    return 0;
}
```

Microsoft Visual Studio
s=0
s=1
s=3
s=6

```
#include <iostream>
using namespace std;

int main()
{
    int i, s ;
    for (i = 1, s = 0; i < 5; s += i++)
        ;
    cout << "s=" << s << endl;
    return 0;
}
```

Microsoft Visual Studio
s=10

3.2 C/C++程序的基本结构和语句种类

4. 空语句：语句以分号作为结束标志，因此单一的分号“；”也算一条语句，它是条空语句，作用是什么也不执行。在程序中，空语句主要用来做空循环体(扩充新功能)

```
#include <iostream>
using namespace std;
```

```
int main()
{
    int i, s ;
    for (i = 1, s = 0; i < 5; s += i++)
        cout << "s=" << s << endl;
    return 0;
```

//虽然cout形式上没有缩进，但仍然是for的子语句，因此循环（输出）四次

C# Microsoft Visual

```
s=0
s=1
s=3
s=6
```

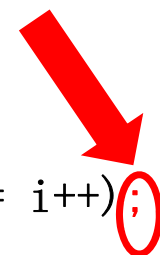
```
#include <iostream>
using namespace std;
```

```
int main()
{
    int i, s ;
    for (i = 1, s = 0; i < 5; s += i++)
        cout << "s=" << s << endl;
    return 0;
```

//虽然cout形式上有缩进，但它不是for的子语句，在for循环结束后才执行一次

C# Microsoft Visual

```
s=10
```



3.2 C/C++程序的基本结构和语句种类

5. 复合语句

- 定义：将多个语句用一对大括号包围，构成一个复合语句，在语法上相当于一个语句
- 表达形式： {语句1; 语句2; 语句3; ……语句n;}
- 注意：
 - 控制语句的子句只能是一条语句，如果一条语句不能完成操作，即需要同时控制多条语句时，可使用复合语句
 - 无论复合语句有多复杂，都被视为一个语句

例：比较a和b两个数值的大小，使max获得两个数中的较大值，而min获得较小值

```
int main()
{
    int a, b, max, min;
    cin>>a>>b;
    if (a>b){
        max = a ;
        min = b ;
    }
    else {
        max = b ;
        min = a ;
    }
    cout<<max<<' '<<min<<endl;
    return 0;
}
```

Microsoft Visual Studio 调试控制台

```
10 20
20 10
```

Microsoft Visual Studio 调试控制台

```
20 10
20 10
```

例：比较a和b两个数值的大小，使max获得两个数中的较大值，而min获得较小值

```
int main()
```

```
{
```

```
    int a, b, max, min;
```

```
    cin >> a >> b;
```

```
    if (a > b) {
```

```
        max = a;
```

```
        min = b;
```

```
    }
```

```
    else {
```

```
        max = b;
```

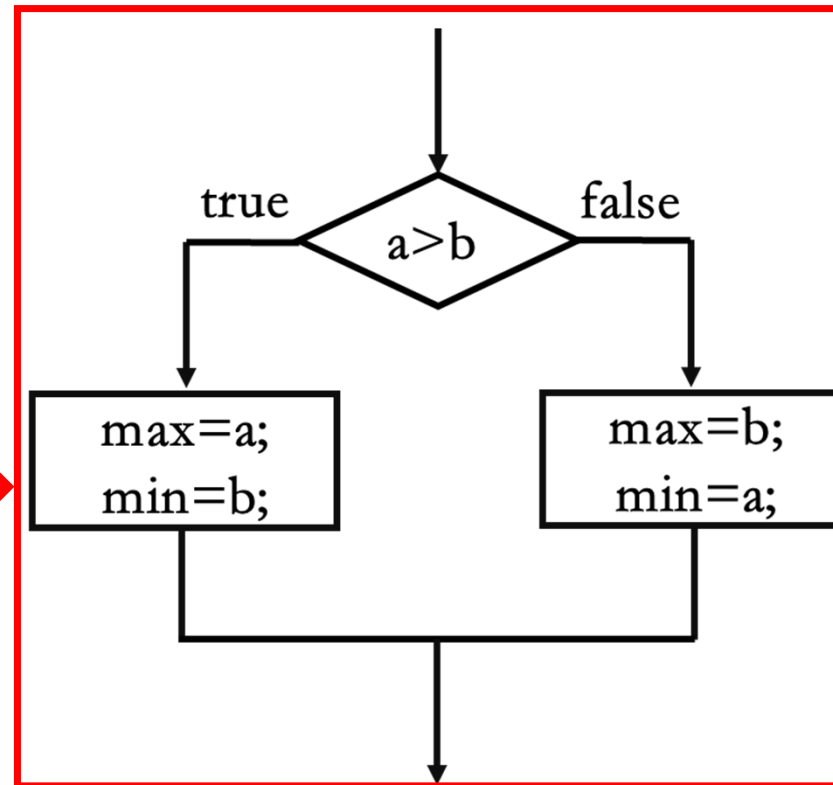
```
        min = a;
```

```
    }
```

```
    cout << max << " " << min << endl;
```

```
    return 0;
```

```
}
```



Microsoft Visual

```
20 10  
20 10
```

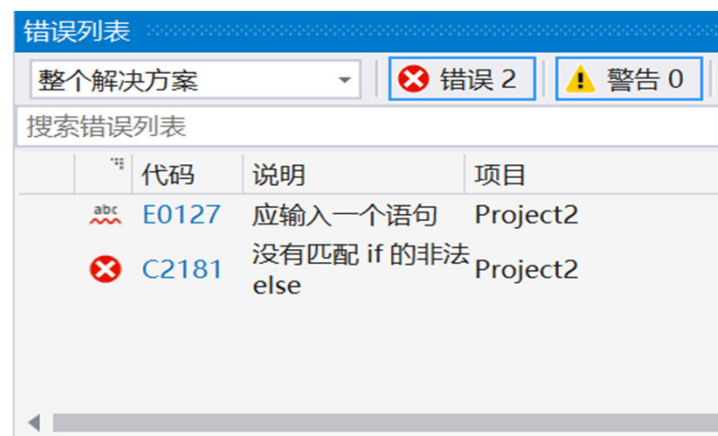
Microsoft Visual S

```
10 20  
20 10
```


3.2 C/C++程序的基本结构和语句种类

```
int main()
{
    int a, b, max, min;
    cin >> a >> b;
    if (a > b) //{
        max = a;
        min = b;
    //}
    else {
        max = b;
        min = a;
    }
    cout << max << " " << min << endl;
    return 0;
}
```

没有大括号?



3.2 C/C++程序的基本结构和语句种类

```
int main()
{
    int a, b, max, min;
    cin >> a >> b;
    if (a > b) // {
        max = a;
        min = b;
    // }
    else {
        max = b;
        min = a;
    }
    cout << max << ' ' << min << endl;
    return 0;
}
```

没有大括号?

```
if (a > b)
    max = a;
```

语句1

```
min = b;
```

语句2

```
else {
    max = b;
    min = a;
}
```

语句3

3.2 C/C++程序的基本结构和语句种类

```
int main()
{
    int a, b, max, min;
    cin >> a >> b;
    if (a > b) // {
        max = a;
        min = b;
    // }
    else {
        max = b;
        min = a;
    }
    cout << max << " " << min << endl;
    return 0;
}
```

没有大括号?

if (a > b)
max = a;

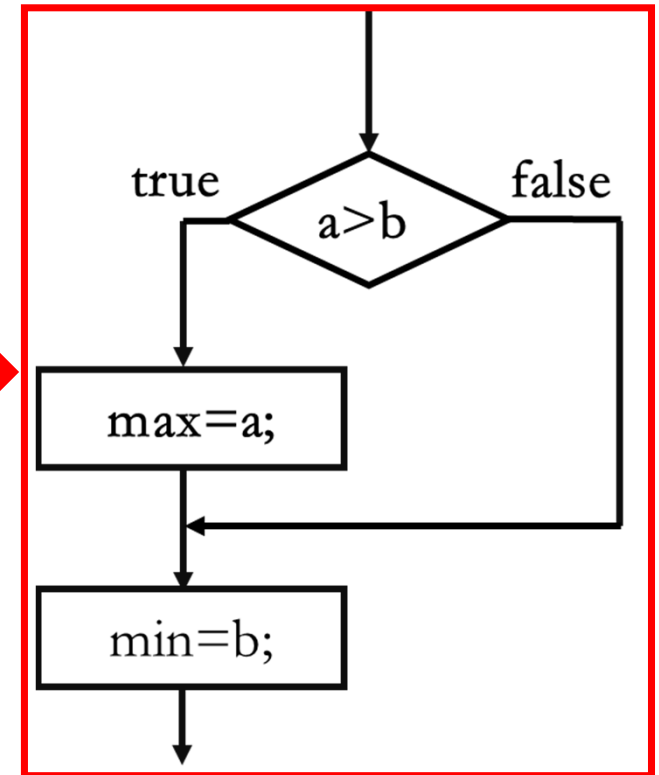
语句1

min = b;

语句2

else {
max = b;
min = a;
}

语句3



```
#include <iostream>
using namespace std;
int main()
{
    int a, b, max, min;
    cin>>a>>b;
    if (a>b){
        max = a ;
        min = b ;
    }
    else //{
        max = b ;
        min = a ;
    //}
    cout<<max<<' '<<min<<endl;
    return 0;
}
```

没有大括号?

Microsoft Visual Studio 调试控制台

```
10 20
20 10
```

Microsoft Visual Studio 调试控制台

```
20 10
20 20
```

```

#include <iostream>
using namespace std;
int main()
{
    int a, b, max, min;
    cin>>a>>b;
    if (a>b){
        max = a ;
        min = b ;
    }
    else //{
        max = b ;
        min = a ;
    //}
    cout<<max<<' '<<min<<endl;
    return 0;
}

```

没有大括号?



```

#include <iostream>
using namespace std;
int main()
{
    int a, b, max, min;
    cin>>a>>b;
    if (a>b){
        max = a ;
        min = b ;
    }
    else
        max = b ;
    min = a ;
    cout<<max<<' '<<min<<endl;
    return 0;
}

```

语句1

语句2

```
if (a>b){
```

```
    max = a ;
```

```
    min = b ;
```

```
}
```

```
else
```

```
    max = b ;
```

```
min = a ;
```

```
cout<<max<<' '<<min<<endl;
```

语句1

语句2

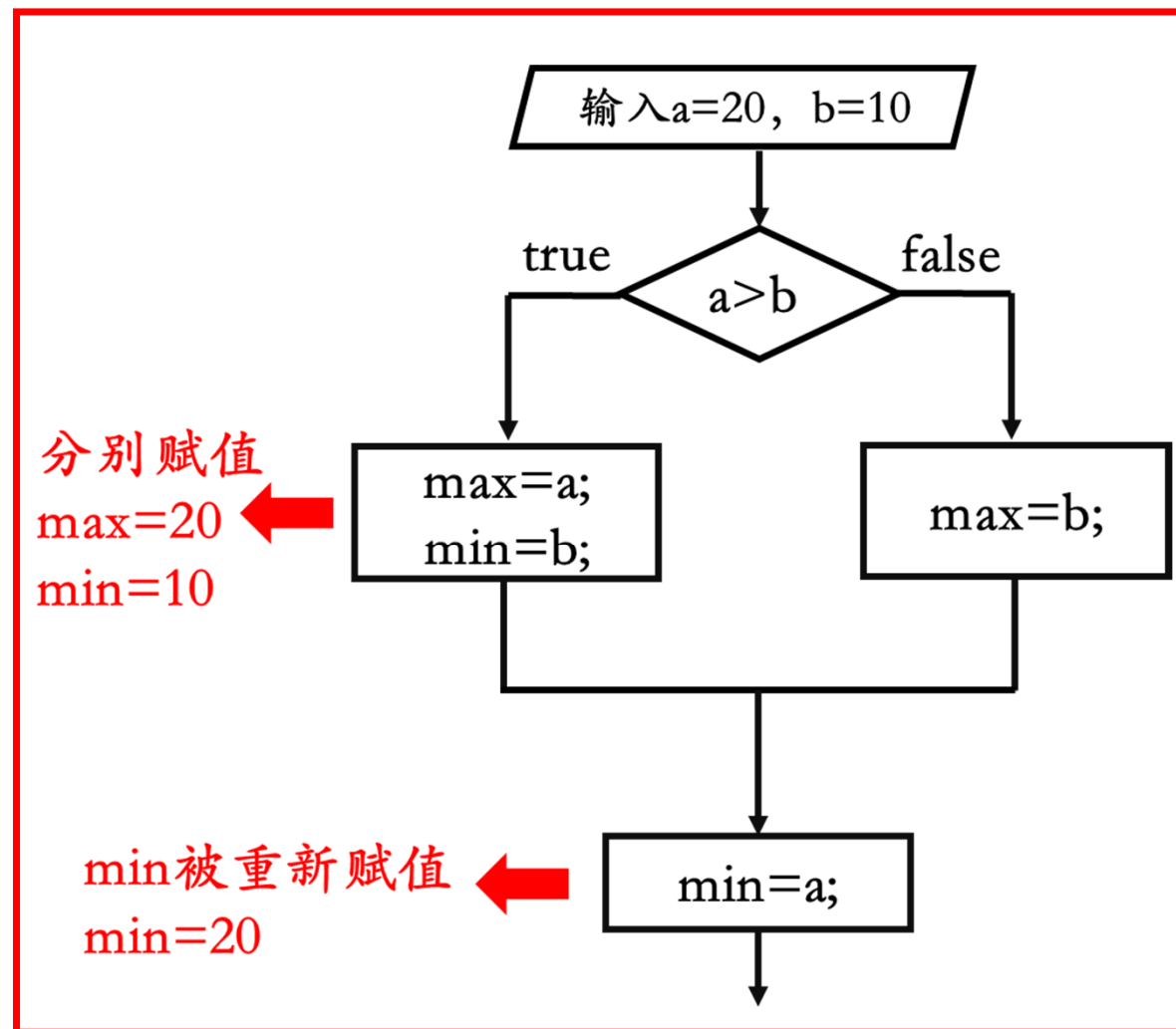
分别赋值

max=20

min=10

min被重新赋值

min=20



Microsoft Vis

```
20 10  
20 20
```

3.3 C++的输入与输出

• 3.31 输入输出的含义

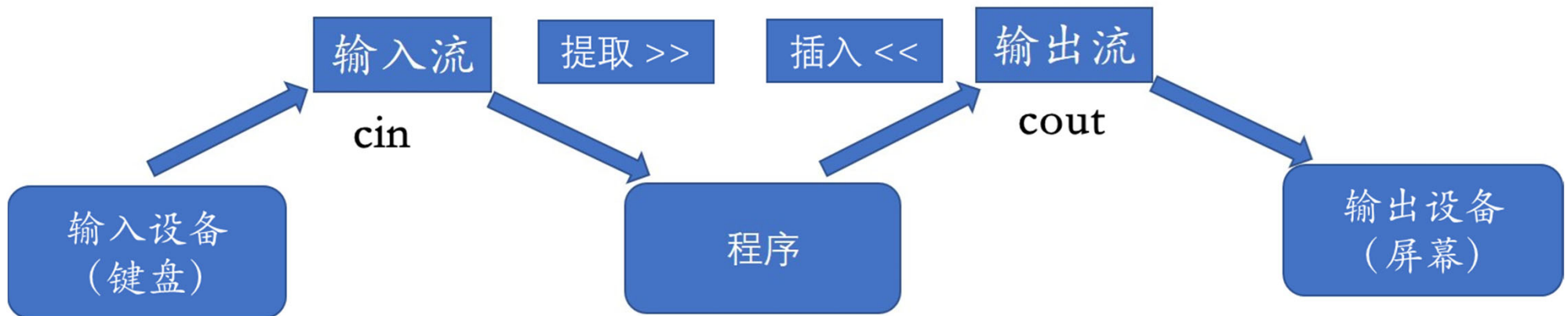
- 从操作系统角度来看，每一个与主机相连的输入输出设备都看作一个文件
- 当程序与外界环境进行信息交换时，存在着两个对象，一个是程序中的对象，另一个是文件对象
- 程序的输入指的是从输入文件将数据传送到程序的过程
- 程序的输出指的是从程序将数据传送给输出文件的过程
- C通过函数实现输入输出—scanf和printf
- C++通过调用输入输出流库中的流对象—cin和cout实现输入输出

3.3 C++的输入与输出

- 3.32 C++的输入输出流

- C++的输入输出是用流的方式实现的
- 输入输出是数据在源端和目的端传送的过程，C++把这个过程称为流
- 数据流由若干个字节组成的字节序列，这些字节按照顺序排列从一个对象传送到另一个对象
- 键盘输入整数123，计算机内部将其转换成整数的二进制补码，输出时再将二进制补码转换成123三个字符在屏幕上输出

3.3 C++的输入与输出



- 输入时，程序从输入流中提取字节；输出时，程序将字节插入到输出流中
- 插入运算符 (<<) 将程序的输出的内容插入到输出流中，在输出设备上输出
- 提取运算符 (>>) 从输入设备的输入流中提取内容送到程序中

3.3 C++的输入与输出

根据输入输出文件的不同，C++的输入输出分为以下三类：

- (1) 对系统指定的标准设备的输入和输出。从键盘输入数据，输出到显示屏。称为标准的输入输出，简称**标准I/O**
- (2) 以外存文件为对象进行输入输出，从外存文件输入数据，数据输出到外存文件，称为文件的输入输出，简称文件I/O
- (3) 对内存中指定的空间进行输入和输出。通常指定一个字符组作为存储空间。称为字符串输入输出，简称串I/O

3.3 C++的输入与输出

3.3.3 标准输出流

- 定义： 利用插入运算符“<<”将输出的内容插入到输出流，在标准输出设备上输出
- 格式： `cout<<表达式1<<表达式2<<……表达式n;`
- 注意：
 1. 插入的数据存储在缓冲区中，不是立即输出，要等到缓冲区满或者碰到换行符（“\n”/endl）才会一起输出
 2. 一个cout语句可写为若干行，或者若干语句
 3. 一个插入运算符只能输出一个值

1. 插入的数据存储在缓冲区中，不是立即输出，要等到缓冲区满或者碰到换行符（“\n” / endl）才会一起输出

```
cout << "hello" << endl;  
cout << "hello" << "\n";  
cout << "hello" << '\n';  
cout << "hello\n";
```

换行符的多种形式



Microsoft Visual Studio 调试控制台

hello

3.3 C++的输入与输出

2. 一个cout语句可写为若干行，或者若干语句

```
cout << "This is a C++ program." << endl;
```

作业要求用第一种方式!

```
cout << "This is " << "a C++ " << "program." << endl;
```

```
cout << "This is "  
    << "a C++ "  
    << "program."  
    << endl;
```

一个语句分4行
前三行无分号

```
cout << "This is ";  
cout << "a C++ ";  
cout << "program.";  
cout << endl;
```

4个语句
每行有分号

3.3 C++的输入与输出

2. 一个cout语句可写为若干行，或者若干语句

```
cout << "This is a C++ program." << endl;
```

```
cout << "This is " << "a C++ " << "program." << endl;
```

```
cout << "This is "  
    << "a C++ "  
    << "program."  
    << endl;
```

一个语句分4行
前三行无分号

```
cout << "This is ";  
cout << "a C++ ";  
cout << "program.";  
cout << endl;
```

4个语句
每行有分号


多行输出时，由于字符串最后没有换行符，因此下一行字符紧接着上一行末尾输出

3.3 C++的输入与输出

3. 一个插入运算符只能输出一个值

例：int a=10, b=15, c=20;

希望输出a, b, c的值，则可以表达为：

1. cout << a << b << c; (正确)  输出结果：？

2. cout << a, b, c; (错，得不到预期结果)

3.3 C++的输入与输出

```
#include <iostream>
using namespace std;

int main()
{
    int a=10, b=15, c=20;
    cout << a << b << c;
    return 0;
}
```

 Microsoft Visual

101520

3.3 C++的输入与输出

```
#include <iostream>
using namespace std;

int main()
{
    int a=10, b=15, c=20;
    cout << a << b << c;
    return 0;
}
```

cout只输出最原始的数据，其它内容都需要自行加入



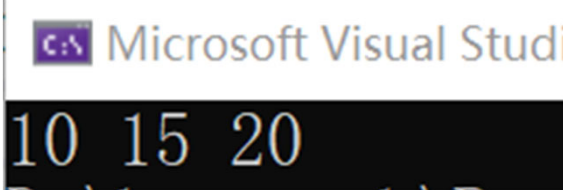
Microsoft Visual

101520

3.3 C++的输入与输出

```
#include <iostream>
using namespace std;

int main()
{
    int a=10, b=15, c=20;
    cout << a << ' ' << b << ' ' << c;
    return 0;
}
```



Microsoft Visual Studi
10 15 20


在a, b, c中间分别输出空格符

3.3 C++的输入与输出

3. 一个插入运算符只能输出一个值

例：int a=10, b=15, c=20;

希望输出a, b, c的值，则

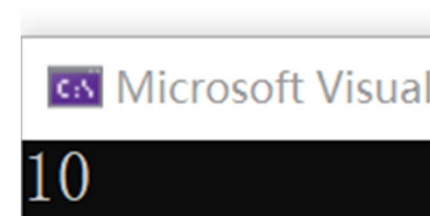
1. cout << a << b << c; (正确)  输出结果：？

2. cout << a, b, c; (错，得不到预期结果)  输出结果：？

3.3 C++的输入与输出

```
#include <iostream>
using namespace std;

int main()
{
    int a=10, b=15, c=20;
    cout << a, b, c;
    return 0;
}
```



3.3 C++的输入与输出

P.849 运算符优先级附录D:

<<和>>的优先级为7，称为按位左移/右移运算符。本章中所称的流插入/流提取运算符本质上是将按位左移/右移运算符经过重载（第11章）而得到的



```
cout << a, b, c;
```

逗号表达式，仅输出a的值

P.849 运算符优先级附录D:

<<和>>的优先级为7，称为按位左移/右移运算符。本章中所称的流插入/流提取运算符本质上是将按位左移/右移运算符经过重载（第11章）而得到的

```
cout << a, b, c << endl;
```

逗号表达式，c << endl 语法错

3.3 C++的输入与输出

3.34 标准输入流

- 定义：利用提取运算符“>>”从输入设备的输入流中提取若干字节送到程序中
- 形式：cin>>变量1>>变量2>>……变量n;
- 注意：
 1. 一行输入内容可分为若干行，或者若干语句
 2. 输入终止条件为回车、空格，输入各个变量时，变量间用空格，回车分隔

3.3 C++的输入与输出

1. 一行输入内容可分为若干行，或者若干语句：

```
cin >> a >> b >> c >> d;
```

```
cin >> a  
    >> b  
    >> c  
    >> d;
```

```
cin >> a;  
cin >> b;  
cin >> c;  
cin >> d;
```


3.3 C++的输入与输出

1. 一行输入内容可分为若干行，或者若干语句：

```
cin >> a >> b >> c >> d;
```

```
cin >> a  
    >> b  
    >> c  
    >> d;
```

```
cin >> a;  
cin >> b;  
cin >> c;  
cin >> d;
```

作业中第二种不准使用，第一种和第三种推荐第一种！

3.3 C++的输入与输出

2. 输入终止条件为回车、空格、非法输入

```
#include <iostream>
using namespace std;
int main()
{
    int i;
    cin >> i;
    cout << i<< endl;
    return 0;
}
```

输入: 123↵	123
输入: 123 456↵	123
输入: 123m ↵	123
输入: ↵	继续等待输入

3.4 C++的格式化输入与输出

- 在输入输出数据时，系统根据数据的类型采取默认的格式（非格式化输入输出），有时在输入输出时需要按指定的格式输出（输出时规定字段宽度，只保留两位小数，数据左右对齐）
- C++用格式操纵符控制格式化输入输出（作业文档中已给出）
- 操纵符是在头文件*iomanip*中定义的，如果使用了操纵符，在程序的开头除了要加*iostream*头文件外，还要加*iomanip*头文件

3.4 C++的格式化输入与输出

```
#include <iostream>
#include <iomanip> // 不要忘记包含此头文件
using namespace std;
int main()
{
    int a;
    cout << "input a:";
    cin >> a;
    cout << a << endl; // 默认以十进制形式输出整数a
    cout << "oct:" << oct << a << endl; // 设置以八进制形式输出整数a
    cout << "hex:" << hex << a << endl; // 设置以十六进制形式输出整数a
    return 0;
}
```

Microsoft Visual

```
input a:21
21
oct:25
hex:15
```

3.4 C++的格式化输入与输出

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    cout << setiosflags(ios::right) << setfill('*') << setw(10) << "12345" << endl; /*设置输出
右对齐，字段宽度为10,输出字符串,空白处以 ‘*’ 填充*/
    cout << resetiosflags(ios::right); //终止已设置的输出格式状态，使ios::right无效
    cout << setiosflags(ios::left) << setfill('*') << setw(10) << "12345 " << endl; /*设置输出
左对齐，字段宽度为10,输出字符串,空白处以 ‘*’ 填充*/
    return 0;
}
```

Microsoft Visual Stu

```
*****12345
12345*****
```

3.4 C++的格式化输入与输出

```
#include <iostream>
```

```
#include <iomanip>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    cout << setiosflags(ios::right) << setfill('*') << setw(10) << "12345" << endl;
```

```
    //cout << resetiosflags(ios::right);
```

```
    cout << setiosflags(ios::left) << setfill('*') << setw(10) << "12345" << endl;
```

```
    return 0;
```

```
}
```

Microsoft Visual Studio 调试

```
*****12345
*****12345
```

此处一定要加上resetiosflag(ios::right)
否则最后一行设置无效

3.4 C++的格式化输入与输出

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    int a=123;
    cout << "012345678901234567890123456789" << endl;
    cout << hex << a << ' ' << a + 1 << '*' << endl; //以十六进制形式输出
    cout << setw(10) << a << ' ' << a + 1 << '*' << endl; //十六进制输出仍然有效
    cout << dec << setw(10) << a << ' ' << setw(10) << a + 1 << '*' << endl;
    return 0;
}
```

setw设置仅一次有效，第二次输出时setw不再生效，按默认格式输出

Microsoft Visual Studio 调试控制台

```
012345678901234567890123456789
7b 7c*
      7b 7c*
    123      124*
```

3.4 C++的格式化输入与输出

```
#include <iostream>
```

```
#include <iomanip>
```

```
using namespace std;
```

```
int main()
```


```
{
```

```
double pi = 3.1415926;
```

```
cout << "pi=" << setiosflags(ios::fixed) << pi << endl; /*设置以固定小  
数位输出，默认输出6位小数*/
```

```
return 0;
```

```
}
```

 Microsoft Visual Studio 调试控制台

```
pi=3.141593
```


3.4 C++的格式化输入与输出

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    double pi = 3.1415926;
    cout << "pi=" << setiosflags(ios::fixed) << pi << endl; /*设置以固定小数位
    输出，默认输出6位小数(精度为6)*/
    return 0;
}
```

如同例3，如果想要改为科学计数法输出，需先要resetiosflags(ios::fixed)

Microsoft Visual Studio 调试控制台

pi=3.141593

3.4 C++的格式化输入与输出

注意:

- 程序在开始必须加`#include <iomanip>`，否则编译出错
- 控制符`setw(n)`只对其后的第一个输出项有效。如果要求在输出数据时都按指定的同一域宽`n`输出，必须在输出每一项前都使用控制符`setw(n)`
- 在用控制符`setiosflags`设置输出格式状态后，如果想改设置为同组的另一状态，应当先`resetiosflags`，终止原来设置的状态，然后再设置其他状态

3.7 顺序结构程序的编写

- 定义：程序设计中最简单，最基本的结构，其特点是程序运行时，按语句的书写次序依次进行

➤ 编写顺序结构程序

例：从键盘输入一个大写字母，要求改为小写字母输出

```
#include <iostream>
using namespace std;
int main()
{
    char c1, c2;//定义两个字符变量
    cin >> c1;//输入一个大写字母
    c2 = c1 + 32;//利用ASCII码转化
    cout << c1 << ' ' << c2 << endl;//输出大小写字母
    return 0;
}
```

- 1、执行时无提示即等待输入
- 2、若输入的不是大写字母出错

➤ 编写顺序结构程序

例：从键盘输入一个大写字母，要求改为小写字母输出

```
#include <iostream>
using namespace std;
int main()
{
    char c1, c2;
    cin >> c1;
    c2 = c1 + 32;
    cout << c1 << ' ' << c2 << endl;
    return 0;
}
```

忘了ASCII码相差32，怎么办？

➤ 编写顺序结构程序

例：从键盘输入一个大写字母，要求改为小写字母输出

```
#include <iostream>
using namespace std;
int main()
{
    char c1, c2;
    cin >> c1;
    c2 = c1 + 32;
    cout << c1 << ' ' << c2 << endl;
    return 0;
}
```

$c2 = c1 + 'a' - 'A';$



► 编写顺序结构程序

例：将用户给出的华氏温度转换成摄氏温度:转换公式为 $c = \frac{5}{9}(f - 32)$

```
#include <iostream>
using namespace std;
int main()
{
    double f, c;
    cout << "请输入华氏温度:";
    cin >> f;
    c = 5.0 / 9 * (f - 32);
    cout << f << " " << c;
    return 0;
}
```

- 1.运算符正确表达形式
- 2.运算符优先级:
(表达式) : 优先级为2
* / : 优先级为5
= : 优先级为16

3.8 字符的输出与输入

3.8.1 用字符输出函数putchar输出字符

- 定义：向显示屏输出一个字符
- 形式： putchar(参数)： 参数可以是一个字符变量/字符常量
- 功能： 向屏幕输出一个字符

```
char a='A';  
putchar(a);           // 变量  
putchar('A') ;       // 常量
```

输出字符'A'


```
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello"<<endl;
    putchar('H');
    putchar('e');
    putchar('l');
    putchar('l');
    putchar('o');
    return 0;
}
```

输出结果:
Hello
Hello

3.8 字符的输出与输入

3.8.2 用于字符输入的getchar函数

- 定义：从键盘上输入一个字符
- 形式：getchar ()
- 不带参数，用来从指定的输入流中提取一个字符，函数的返回值就是读入的字符
- 功能：输入一个字符给指定的变量

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

输入: AA
输出:
A
65



See you @结构体