

If, For, Array

王慧妍

why@nju.edu.cn

南京大学



软件学院



计算机软件研究所



回顾

Variables Constants Data Types

Operators Expressions Assignment Statements

I/O (Input/Output)

本讲

- If
- For
- Array

if条件语句

- 一个基本的if语句有关键字if，跟上在括号里的一个表示条件的逻辑表达式，然后是一堆大括号{}之间的若干条语句。
 - 若表达式成立，则执行后面跟着的语句
 - 若不成立，则不执行后续跟着的语句，若有执行对应else分支后的语句

```
if(...) {  
    ...  
}
```

```
if(...) {  
    ...  
}  
else {  
    ...  
}
```

```
if(total > amount)  
    total += amount + 10;
```

注意，if语句这一行结束并没有分号，后面的语句写在if下一行并且缩进

判断成绩

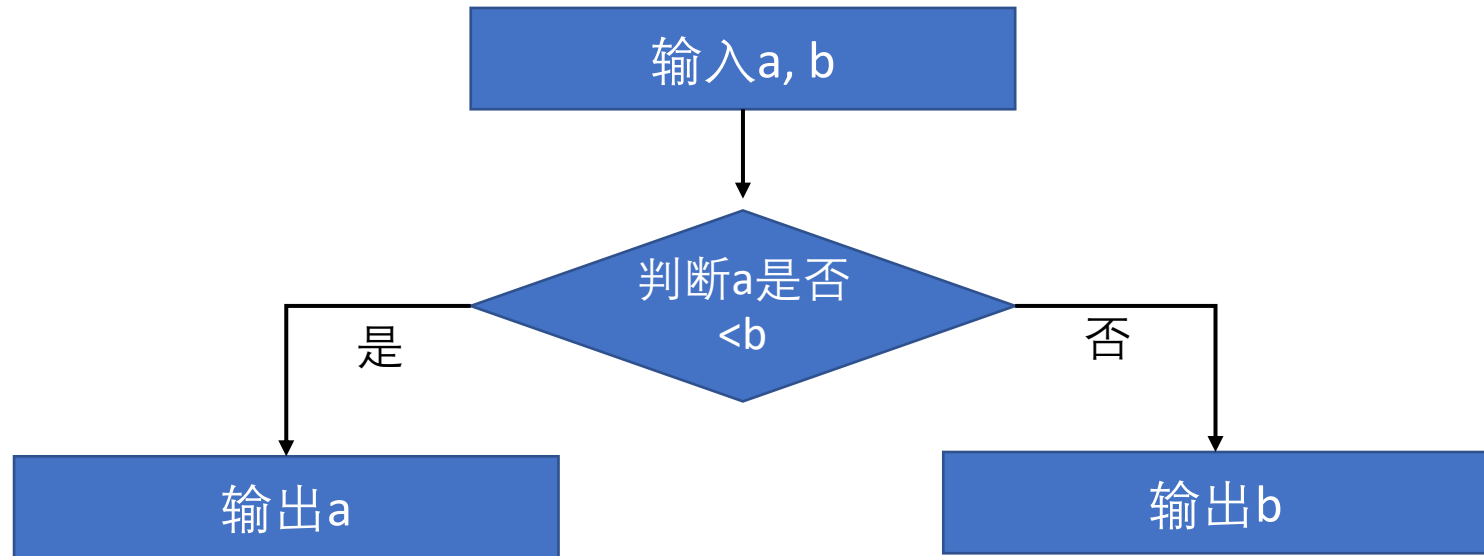
```
#include<stdio.h>
int main(){
    int score;
    const int PASS = 60;
    printf("Please enter a score: ");
    scanf("%d", &score);

    if(score >= PASS)
        printf("Cong! YOU PASS!\n");
    else
        printf("Sorry! YOU FAIL!\n");

    return 0;
}
```

MIN

- $\text{result} = \min\{a, b\}$



if语句常见错误

- 忘记大括号
- `if(;;`
- 错误使用`==`和`=`
- 代码风格和合适的缩进

if条件语句

- else总是匹配前面最近的if
 - 无大括号隔开
- 缩进并不能暗示匹配情况
 - 新手Tips：始终加上{}，自我检查后的缩进style

```
if(a<b)//<=
|   if(a<10)
|       printf("min1: %d\n", a);
else
|   printf("min2: %d\n", b);
```


MIN

- $\text{result} = \min\{a, b, c\}$
- min.c



```
int min;  
if (a > b) {  
    if (b > c) {  
        min = c;  
    } else {  
        min = b;  
    }  
} else {  
    if (a > c) {  
        min = c;  
    } else {  
        min = a;  
    }  
}
```

级联的if

```
if(...) {  
    ...  
}  
else{  
    if(){  
        ...  
    }  
    else(){  
        ...  
    }  
}
```



```
if(...) {  
    ...  
}  
else if(){  
    ...  
}  
else(){  
    ...  
}
```

$\min\{a, b, c\}$

- [min.c](#)



逻辑表达式

- 关系运算符

- $<$, $>$, $<=$, $>=$

- (优先级低于算术运算符)

$i + j < j * k$

$(i + j) < (j * k)$

$i > j > k$

$(i > j) > k$

- 判等运算符

- $==$, $!=$

- (优先级低于关系运算符)

$i < j == j < k$

$(i < j) == (j < k)$

- 逻辑运算符

- $\&\&$ (与), $\|\|$ (或), $!$ (非)

- $(i != 0) \&\& (j / i > 0)$

- 注意和位运算区分： $\&$, $|$, \sim

运算符优先级

- $m = a < b$
- $m == a \&\& b$
- $m == a \& b$

表 2-1 C 语言运算符优先级表（由上至下，优先级依次递减）

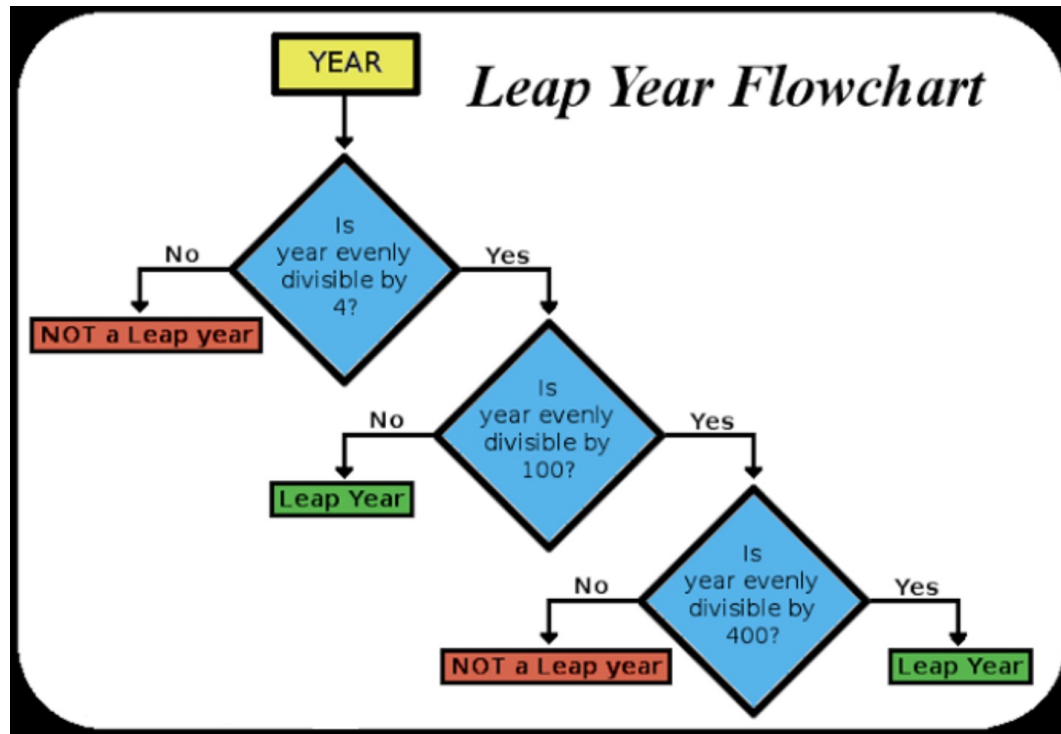
运 算 符	结 合 性
() [] -> .	自左向右
! ~ ++ -- - (type) * & sizeof	自右向左
* / %	自左向右
+ -	自左向右
<< >>	自左向右
< <= > >=	自左向右
= !=	自左向右
&	自左向右
^	自左向右
	自左向右
&&	自左向右
	自左向右
?:	自右向左
assignments	自右向左
,	自左向右

分段函数

- $f(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 2x, & x > 0 \end{cases}$

LEAP year

- 如何判断输入是否是闰年？
 - 4的倍数，且不是100的倍数的，为闰年
 - 400的倍数，为闰年

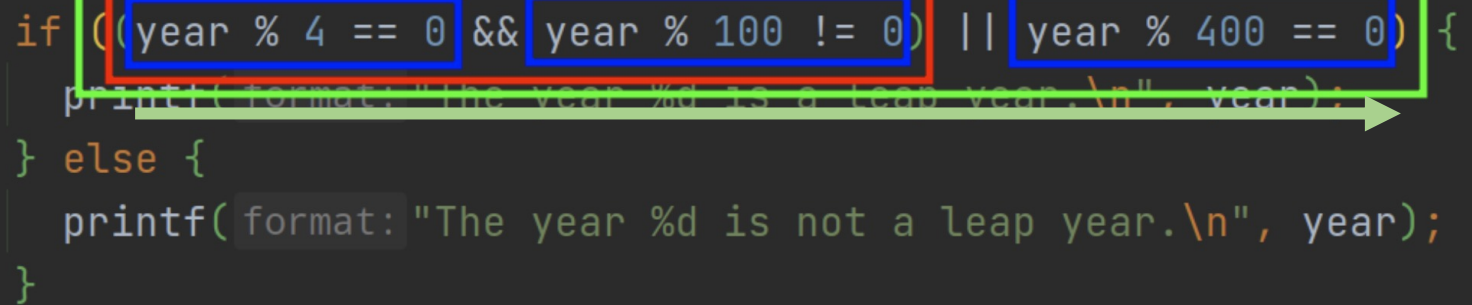


[leap.c](#)


```
if (year % 4 != 0) {  
    printf(format: "The year %d is a common year.\n", year);  
} else if (year % 100 != 0) {  
    printf(format: "The year %d is a leap year.\n", year);  
} else if (year % 400 != 0) {  
    printf(format: "The year %d is a common year.\n", year);  
} else {  
    printf(format: "The year %d is a leap year.\n", year);  
}
```

```
if ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0) {  
    printf(format: "The year %d is a leap year.\n", year);  
} else {  
    printf(format: "The year %d is not a leap year.\n", year);  
}
```

- year = 2001
- year = 2020



```
if ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0) {  
    printf(format: "The year %d is a leap year.\n", year);  
} else {  
    printf(format: "The year %d is not a leap year.\n", year);  
}
```

The image shows a C code snippet for checking if a year is a leap year. The code is displayed on a dark background with syntax highlighting. Annotations include: a red arrow pointing from the first condition to the second; a green box enclosing the entire if statement; a blue box around the first condition; a red box around the second condition; and a green arrow pointing from the first condition to the printf statement.

条件表达式

- 三元运算符

- 表达式1 ? 表达式2 : 表达式3
- 将简单if语句简化为一条表达式

- $i > j ? i : j$
 - 若 $i > j$ 为真，取 i ，否则取 j

- [ternary.c](#)

```
int x = a > b ? a : b;  
int y = (a > 0 ? a : 0) + b;
```

- 回顾优先级
 - $? :$ 优先级高于赋值，低于一般算术运算符

switch

```
switch (expression)
{
case /* constant-expression */:
    /* code */
    break;
case /* constant-expression */:
    /* code */
    break;

default:
    break;
}
```

switch

- 输入1-12数字，输出对应当前月份



[month.c](#)

[month1.c](#)

成绩转换

- 输出成绩的分级
 - A : 90-100
 - B : 80-89
 - C : 70-79
 - D : 60-69
 - F : <60
- 级联的if/switch-case

For循环

- Given a set A of integers, to compute their minimum.



```
for ( <expression> ; <expression> ; <expression> )  
    <statement>
```

循环开始前的准备

循环结束条件

每轮循环的最后
一个执行
惯用法i++

For循环

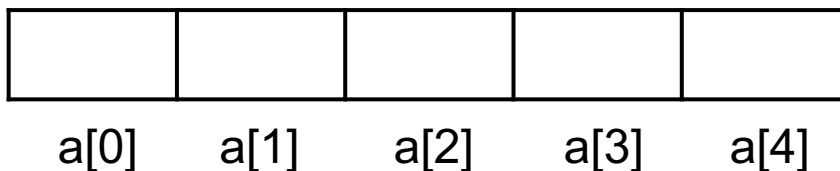
- For循环就像一个计数循环，设定一个计数器，初始化后，在计数器到达某个值之前，重复循环体，每执行一轮，计数器更新
- 一个简单点的任务
 - 输入正整数 n ，计算 $n!$
 - [fact.c](#)

For循环

- For循环就像一个计数循环，设定一个计数器，初始化后，在计数器到达某个值之前，重复循环体，每执行一轮，计数器更新
- 一个简单点的任务
 - 输入正整数n，计算n!
 - [fact.c](#)
- for语句三个表达式的省略
 - [simplefor.c](#)
- `for(int i = 0; i < n; i++)` // int i只在循环里面用到
 - 循环次数
 - i的内外重名？

数组

- 数组是含有多个数据值的数据结构，并且每个数据值拥有同样的数据类型
 - 存取特别的数组元素，可以以取下标的形式读取
 - 元素可以是任意类型，长度为任何常量表达式
 - `int a[5];`
 - `#define M 10; int b [M];`
 - Do not use: `int a[n];` (VLA)



`[]`: *subscript operator* (下标运算符)

数组的定义与初始化

- `#define NUM 5`
- 初始化
 - `int nums[NUM] = {0};`
 - `{0,0,0,0,0}`
 - `int nums[] = {0};`
 - `{0}`
 - `int nums[NUM] = {1};`
 - `{1,0,0,0,0}`
 - `int nums[NUM] = {[2]=1};` //指示器
 - `{0,0,1,0,0}`

数组的定义与初始化

- `#define NUM 5`
- 初始化
 - ~~`int numbers[NUM] = {};`~~
 - `//forbidden by C99, allowed in GCC by default`
 - ~~`int numbers[NUM];`~~
 - `//may contain garbage values`
 - ~~`int numbers[];`~~
 - You must specify the size so that the compiler can allocate memory for it.

几种数组操作惯用用法

- 初始化数组

```
for(i = 0; i < N; i++)  
    a[i] = 0;
```

- 按序输入数组

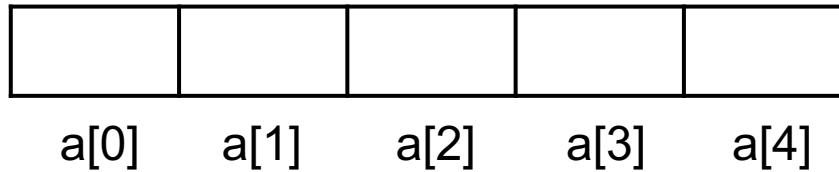
```
for(i = 0; i < N; i++)  
    scanf("%d", &a[i]);
```

- 数组遍历求和

```
for(i = 0; i < N; i++)  
    sum += a[i];
```

Pay attention!

- 长度为n的数组下标总是0到n-1!



```
int a [10], i;  
for(i = 1; i<= 10; i++)  
    a[i] = 0;
```

- Now, given a set A of integers, to compute their minimum.
- [minInArray.c](#)



Star Pyramid

- [stars.c](#)



Keep coding!

下课咯

