中山大學
SUN YAT-SEN UNIVERSITY

Selection Structure (选择结构)

中山大学计算机学院

主讲人：黎卫兵　　中山大学MOOC课程组

C语言的语句（Statements）与复合语句

语句有五种类型：

1) **复合语句**（又称为“块，Block”）
2) 表达式语句
3) 选择语句
4) 循环语句
5) 跳转语句

在复合语句语法 {…}，例如：

```
int main() { // 复合语句的开始
    int n = 1; // 声明，n 是块中自动变量
    n = n + 1; // 表达式语句
    printf("n = %d\n", n); // 表达式语句
    return 0; // 返回语句
} // 复合语句之结尾，函数体之结尾
```

在复合语句中，语句按**顺序执行**。

语句 - cppreference.com

目录

CONTENTS

布尔表达式（Boolean expression）

在现实世界中，需要表达事物（是，否）、（对、错）、（真、假）、（是、非）、（正、反）、（阳、阴）... 等二值语义状态，在数学中映射为（true，false），在计算机内部用（1，0）表示。这些二值对，统称**布尔值**。

在C语言中，返回值属于 {0,1} 的表达式称为**布尔表达式**，包括：

- 比较表达式
- 逻辑表达式。操作数非零则转为1
- 使用 (_Bool) Expr 强制转为数值为布尔值。非零则转为1

C语言对布尔类型的支持

C 语言内置支持：

- 布尔常数：1，0
- 布尔类型：_Bool

通过 stdbool.h 扩展

- 常数：**true** 1; **false** 0;
- 类型：**bool** _Bool;

满足人们的书写习惯

```c
/*boolean constant and type*/
#include<stdio.h>
#include<stdbool.h>

int main() {
    bool x = 7; //相当于 x=(_Bool)7
    bool b = false;
    printf("%x\n",x);
    printf("%u %u\n",sizeof(true),sizeof(b));
    printf("%u %u\n",sizeof(bool),sizeof(_Bool));
}
```

注意：true, false, bool 不是关键字，可以再定义

布尔表达式-案例

该案例要点：

- 产生一个随机数
  - srand（unsigned）设置种子
  - rand() 会随机生成一个位于 0 ~ RAND_MAX 之间的整数
- 判断数是否在一个区间
  - in
  - not in

```
/*boolean expr and random*/
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

int main() {
    //give a random between 0..n-1
    srand((unsigned)time(NULL)); //seed
    const int n = 10;
    int i = rand() % n;

    //print bool expr i in (2,8]
    printf("%d,%u\n", i ,i>2 && i<=8);
    //print bool expr i not in (2,8]
    printf("%d,%u\n", i ,i<=2 || i>8);
    printf("%d,%u\n", i ,!(i>2 && i<=8));
}
```

条件与路径选择

在实际生活中，经常需要根据某个条件是否满足来决定是否执行指定的操作任务。例如，我们用类似 C 语言的方式来描述一个指路的程序：

```
go_ahead(200m)
if (meet the cross?) then
    turn(left)
```

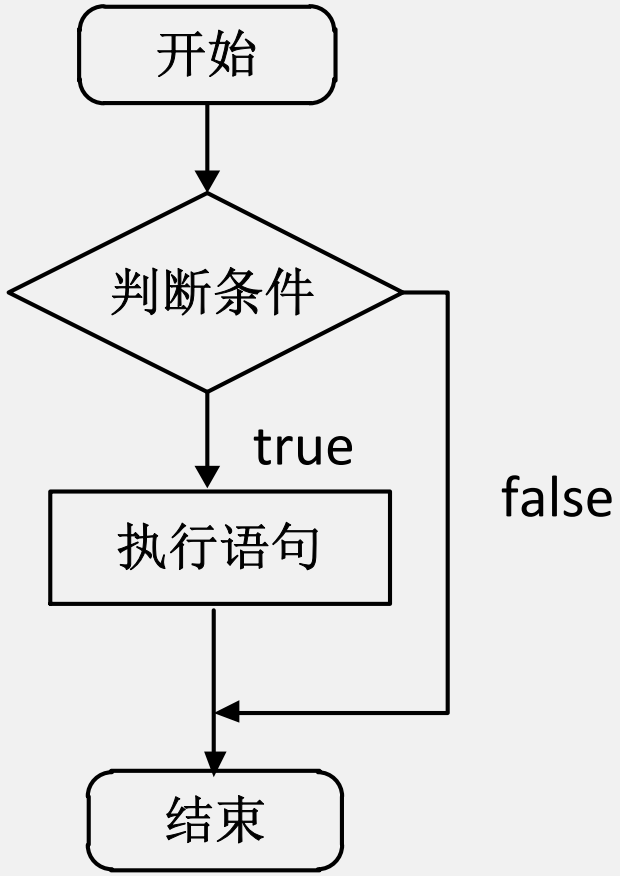
遇到路口，才能选择执行左转指令。显然，if 语句自然易于理解。

后面，我们研讨**单路选择**，**两路选择**，**多路选择**的语法和应用场景。

if语句 - 单路选择

```
if(表达式)
    语句1
```

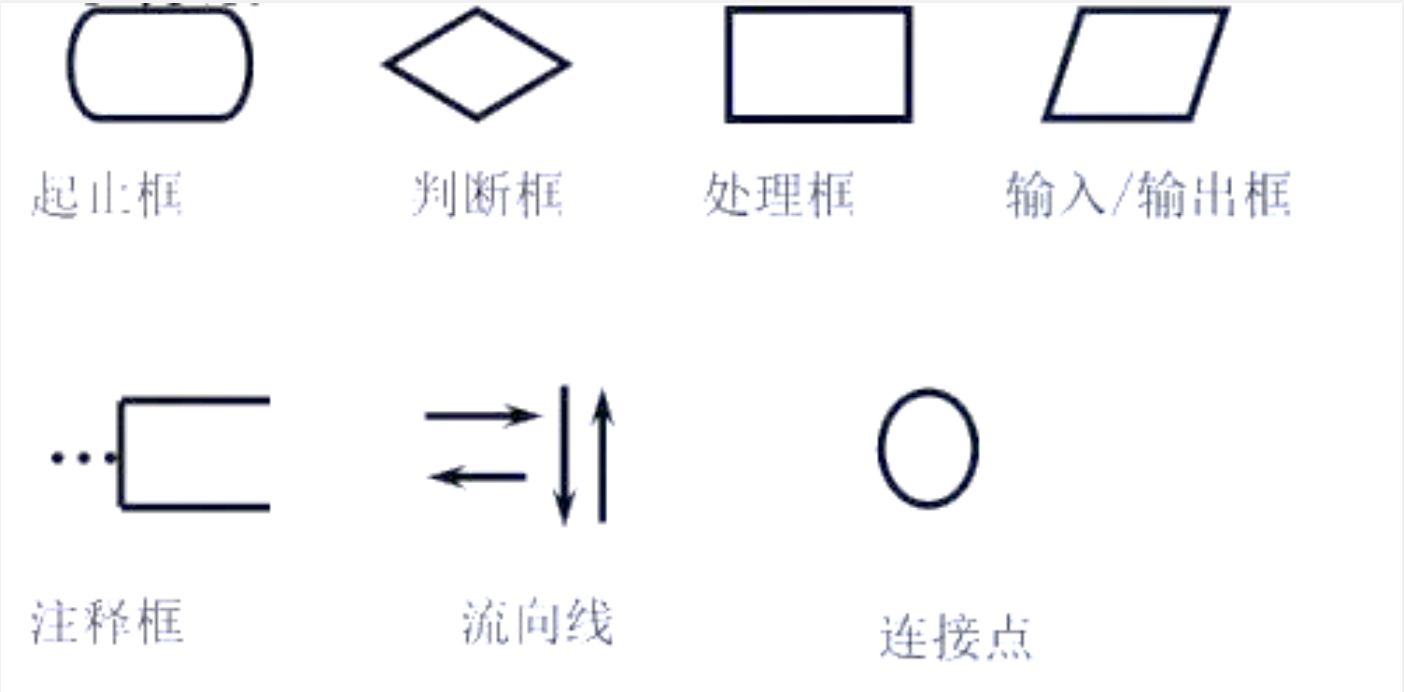
- If 语句中的“表达式”是布尔表达式。
  - 表达式必须在括号之中；
  - 表达式不是布尔表达式则发生强制转换
  - 传统上，非零值表示条件成立
- “语句”是任意一种语句（不包括申明）
  - 复合语句后面不一定需要分号
  - 表达式语句等必须分号结束



流程图

程序流程图（FlowChart）

流程图（FlowChart）是表示算法、工作流或流程的一种框图表示。



if语句 - 单路选择 – 案例

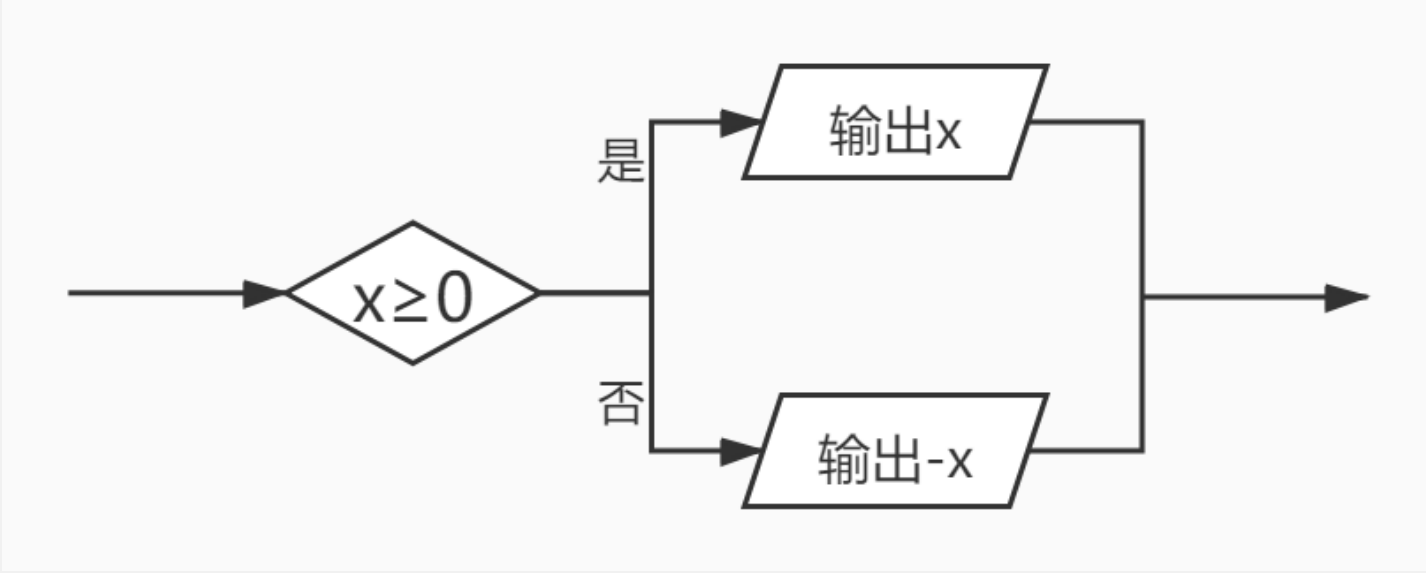
```
/*oneway if - compute area*/
#include<stdio.h>

int main() {
        double radius, area;
        const double PI = 3.14;

        printf("input radius:");
        scanf("%lf",&radius);

        /*compute area*/
        area = 0;                          如果没有这句，程序会出现什么问题?
        if (radius > 0)
           area = radius * radius * PI;

        printf("the area for the circle of redius %lf is %lf",
                radius, area);
}
```

if语句 – 双路选择

输出一个数x的绝对值。

根据这个数x的正负这个条件进行判断，如果满足x≥0的条件，我们输出x；否则，输出-x。

```
if(x >= 0)
     printf("%d", x);
else
     printf("%d", -x);
```



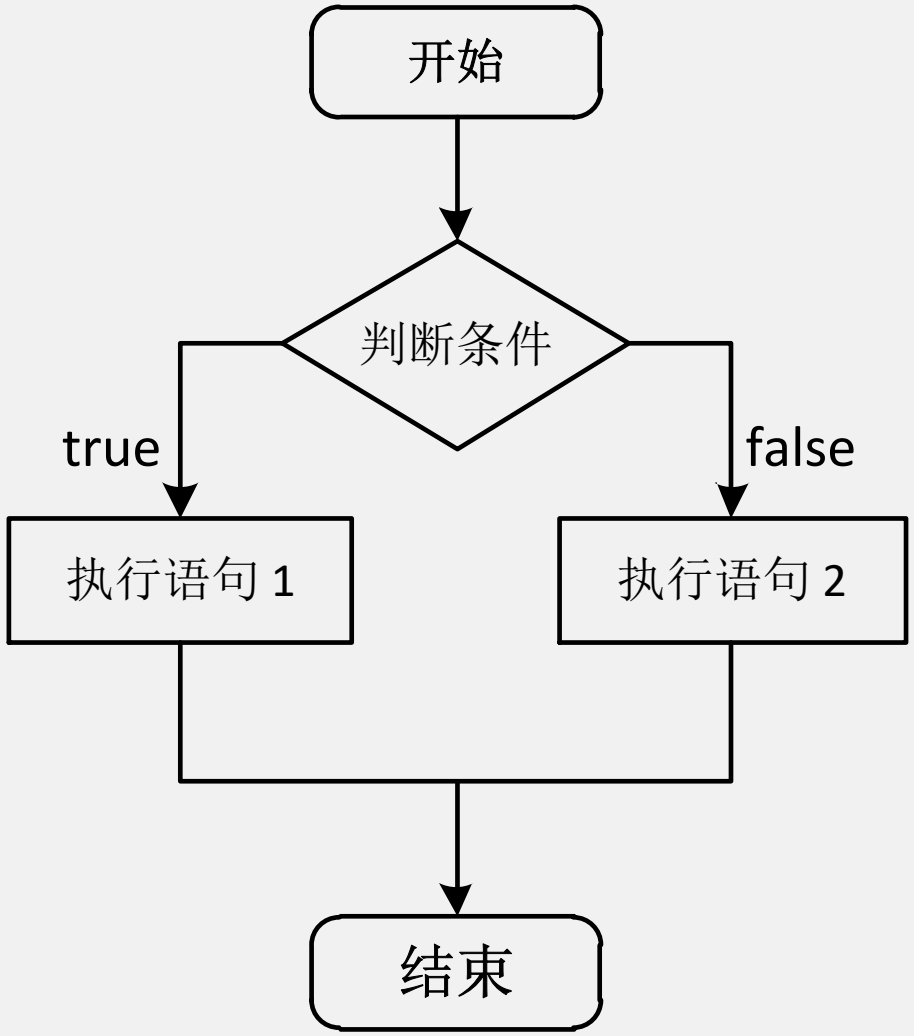
if语句 – 双路选择 – 语法

```
if(表达式)
        语句1
else
        语句2
```

- 该if语句意为，如果表达式的值为真，那么执行下面的语句1；否则执行下面的语句2

if语句 – 双路选择 – 语法

```
if(表达式)
        语句1
else
        语句2
```



if语句 – 双路选择 – 实现的多样性

```
if(x ≥ 0)
      printf("%d", x);
else
      printf("%d", -x);
```

if语句 – 双路选择 –实现的多样性

```
if(x ≥ 0)
      printf("%d", x);
else
      printf("%d", -x);
```

```
if(x ≥ 0){
      printf("%d", x);
}else{
      printf("%d", -x);
}
```

if语句 – 双路选择 –实现的多样性

```
if(x ≥ 0)
      printf("%d", x);
else
     printf("%d", -x);
```

```
if(x ≥ 0)
      printf("%d", x);
if(x < 0)
     printf("%d", -x);
```

if语句 – 双路选择 – 案例研究

求两个数中的较大值

```
/*two way if max*/
#include <stdio.h>
int main()
{
    int a, b, max;
    printf("输入两个整数: ");
    scanf("%d%d", &a, &b);

    //请将双路选择改为单路选择
    if(a>b) max=a;
    else max=b;

    printf("%d和%d的较大值是: %d\n", a, b, max);
    return 0;
}
```

if语句 – 练习

应用：教小朋友学数学

```
/*ex-add-learning*/
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

int main() {
    //give two random between 0..9
    srand((unsigned)time(NULL)); //seed

    int i,j,sum;
    i = rand() % 10;
    j = rand() % 10;

    printf("%d + %d = ",i,j);
    scanf("%d",&sum);

    //请根据学生的回答，分别使用以下语句
    printf("恭喜你，答对了！\n");
    printf("Oooops，再努力一次。\n");
}
```

if语句 – 多路选择 – 语法

```
if(表达式1)
        语句1
else if(表达式2)
        语句2
else if(表达式3)
        语句3
...
else if(表达式n)
        语句n
else
        语句n+1
```

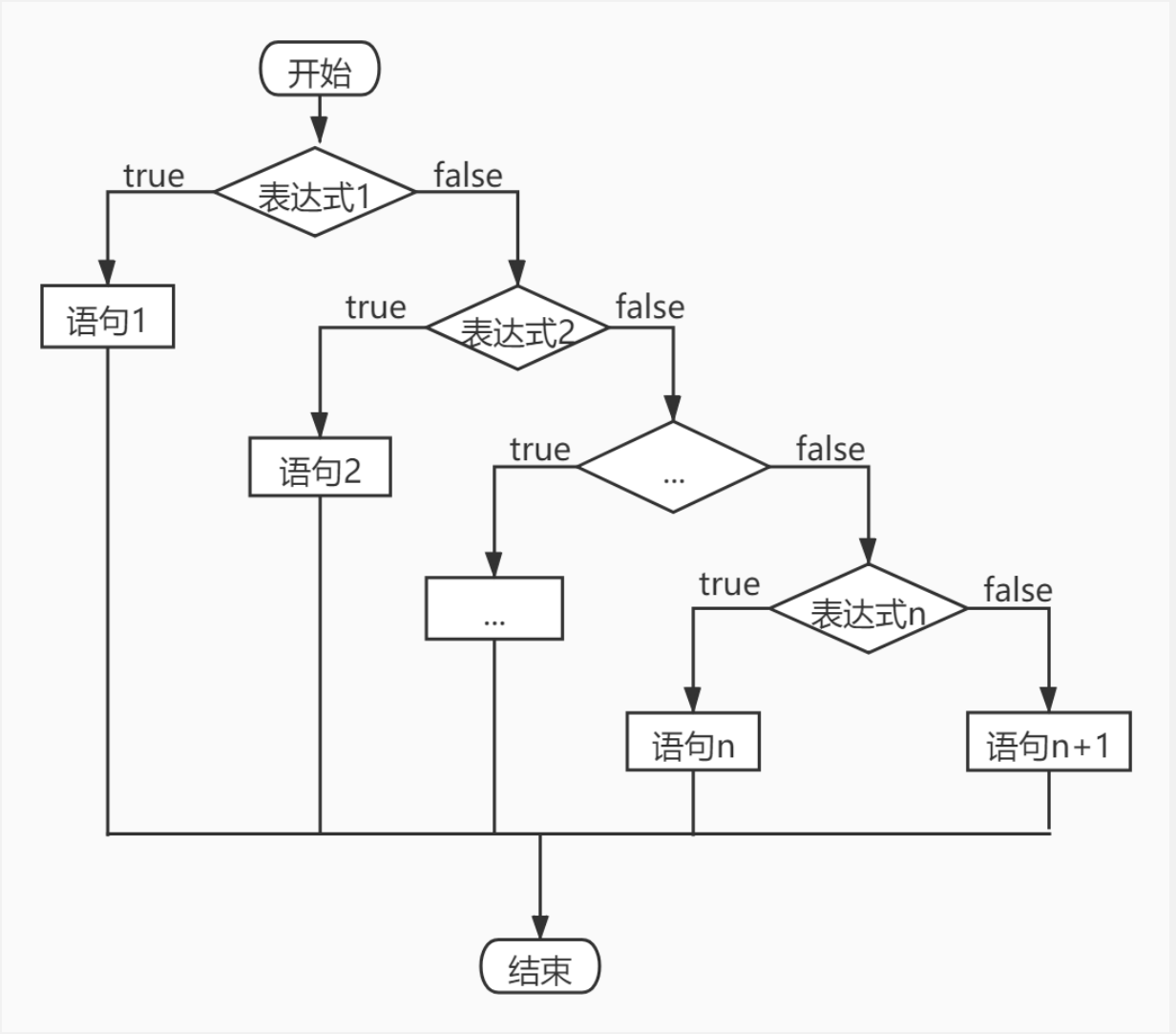
该选择结构意为，首先判断表达式1的值是否为真，如果为真，那么执行下面的语句1，之后跳过剩下的判断；

否则，判断表达式2的值是否为真，如果为真，那么执行下面的语句2，之后跳过剩下的判断；

继续判断下去...直到找到一个表达式的值为真，执行后面的语句；否则在最后的else分支中，执行语句n+1

if语句 – 多路选择 – 语法

```
if(表达式1)
        语句1
else if(表达式2)
        语句2
else if(表达式3)
        语句3
...
else if(表达式n)
        语句n
else
        语句n+1
```



if语句 – 多路选择 – 应用

```
int x = 500;
if(x < 10)
      printf("个位数");
else if(x < 100)
      printf("十位数");
else if(x < 1000)
      printf("百位数");
```

该选择结构下，首先判断x < 10这个条件是否满足，如果满足则输出；不满足则继续判断下一个条件，直到碰到一个满足的条件执行其后的语句、或者所有的条件都不满足不执行任何语句。

该程序执行下列语句：

```
printf("百位数");
```

问题：第一个 if 条件满足后，执行 printf(“十位数”); 是否合适 ？如何在多个结果选择第一个执行的结果

嵌套if语句

```
if(表达式1)
{
    if(表达式2)
        语句1
    else
        语句2
}
else
{
    ...
}
```

语句块1

语句块2

- 在if语句中又包含了一个或多个if语句称为if语句的嵌套。
- 在该选择结构中，首先判断表达式1是否为真，如果为真，进入其后的语句块1中；不然进入语句2中。
- 假设进入语句1中，此时碰到了嵌套if结构，此时可视为碰到了一个单独的if结构，继续判断表达式2是否为真，如果为真，执行其后的语句1，否则执行语句2

嵌套if语句

```
int x = 3;
if(x > 0)
{
    if(x % 2)
        printf("x是正奇数");
    else
        printf("x是正偶数");
}
else
{
    if(x % 2)
        printf("x是负奇数");
    else
        printf("x是负偶数");
}
```

嵌套if语句

```
int x = 3;
if(x > 0)
{
    if(x % 2)
        printf("x是正奇数");
    else
        printf("x是正偶数");
}
else
{
    if(x % 2)
        printf("x是负奇数");
    else
        printf("x是负偶数");
}
```

在该例子中首先判断条件 x>0 是否成立，如果成立，执行第一个语句块，不然执行else后面的语句块。

由于x = 3，条件x > 0判断为真，进入第一个语句块。继续判断x % 2 == 1是否为真，此时判断为真，并输出"x是正奇数"

```
printf("x是正奇数");
```

嵌套if语句

```
if(a!=b)  // ①
if(a>b) printf("a>b\n");  // ②
else printf("a<b\n");  // ③
```

嵌套if语句

```
if(a!=b)  // ①
if(a>b) printf("a>b\n");  // ②
else printf("a<b\n");  // ③
```

```
if(a!=b) {
    if(a>b) printf("a>b\n");
    else printf("a<b\n");
}
```

C语言规定，else 总是与它前面最近的 if 配对

嵌套if语句

```
int a = 1, b = 3, c = 5, d = 4, x;
if (a < b)
if (c<d)
x = 1;
else
if (a<c)
if (b<d)
x = 2;
else
x = 3;
else
x = 6;
else
x = 3;
```

嵌套if语句

```
int a = 1, b = 3, c = 5, d = 4, x;
if (a < b)
if (c<d)
x = 1;
else
if (a<c)
if (b<d)
x = 2;
else
x = 3;
else
x = 6;
else
x = 3;
```

```
int a = 1, b = 3, c = 5, d = 4, x;
if (a < b)
   if (c<d)
      x = 1;
   else
      if (a<c)
         if (b<d)
            x = 2;
         else
            x = 3;
      else
            x = 6;
else
   x = 3;
```

嵌套if语句

Google的代码风格规范

只使用空格，每次缩进2个空格。不要在代码中使用tabs。

不在圆括号内使用空格. 关键字 if 和 else 另起一行

```
if (condition) {  // 圆括号里没有空格.
  ...  // 2 空格缩进.
} else if (...) {  // else 与 if 的右括号同一行.
  ...
} else {
  ...
}
```

嵌套if语句

Google的代码风格规范

增强可读性，简短的条件语句允许写在同一行。
只有当语句简单并且没有使用 else 子句时使用。

```
if (x == kFoo) return new Foo();
if (x == kBar) return new Bar();
```

嵌套if语句

Google的代码风格规范

如果语句中某个 if-else 分支使用了大括号的话，其它分支也必须使用。

```
// 不可以这样子 - IF 有大括号 ELSE 却没有.
if (condition) {
  foo;
} else
  bar;

// 不可以这样子 - ELSE 有大括号 IF 却没有.
if (condition)
  foo;
else {
  bar;
}
```

嵌套if语句

Google的代码风格规范

如果语句中某个 if-else 分支使用了大括号的话，其它分支也必须使用。

```
// 只要其中一个分支用了大括号，两个分支都要用上大括号.
if (condition) {
  foo;
} else {
  bar;
}
```

switch语句

```
switch(表达式)
{
    case 常量1: 语句1; break;
    case 常量2: 语句2;
    case 常量3: 语句3; break;
    ...
    case 常量n: 语句n; break;
    default: 语句n+1
}
```

- 相比if语句，switch语句常常用来实现多分支选择结构。因为多层if语句的可读性通常没有switch结构好。
- switch结构中的”表达式”，其值的类型应为整数类型(包括字符类型char等)
- switch下面的花括号是一个语句块，其中包含了多行以关键字case为开头的语句。每行语句case后面跟一个常量。

switch语句

```
switch(表达式)
{
    case 常量1: 语句1; break;
    case 常量2: 语句2;
    case 常量3: 语句3; break;
    ...
    case 常量n: 语句n; break;
    default: 语句n+1
}
```

执行switch结构时，通常先计算switch后面”表达式”的值，然后将它逐个与case语句中的常量进行比对。

如果某一个case语句中的常量相同，流程就转到此分支、执行此case后面的语句，执行完后由break语句跳出switch结构。

例如，表达式值是常量2，则执行 语句2 和 语句3 再 break

switch语句

```
/*switch weekday*/
#include <stdio.h>

int main(){
    int a;
    printf("Input integer number:");
    scanf("%d",&a);

    switch(a){
        case 1: printf("Monday\n");
        case 2: printf("Tuesday\n");
        case 3: printf("Wednesday\n");
        case 4: printf("Thursday\n");
        case 5: printf("Friday\n");
        case 6: printf("Saturday\n");
        case 7: printf("Sunday\n");
        default:printf("error\n");
    }
    return 0;
}
```

switch语句

```
#include <stdio.h>
int main(){
    int a;
    printf("Input integer number:");
    scanf("%d",&a);
    switch(a){
        case 1: printf("Monday\n"); break;
        case 2: printf("Tuesday\n"); break;
        case 3: printf("Wednesday\n"); break;
        case 4: printf("Thursday\n"); break;
        case 5: printf("Friday\n"); break;
        case 6: printf("Saturday\n"); break;
        case 7: printf("Sunday\n"); break;
        default:printf("error\n"); break;
    }
    return 0;
}
```

break 是C语言中的一个关键字，专门用于跳出 switch 语句。

所谓“跳出”，是指一旦遇到 break，就不再执行 switch 中的任何语句，包括当前分支中的语句和其他分支中的语句；也就是说，整个 switch 执行结束了，接着会执行整个 switch 后面的代码。

switch语句

```
case 10: printf("..."); break;
case 8+9: printf("..."); break;
case 'A': printf("..."); break;
case 'A'+19: printf("..."); break;
case 9.5: printf("..."); break;
case a: printf("..."); break;
case a+10: printf("..."); break;
```

switch语句

```
case 10: printf("..."); break;  //正确
case 8+9: printf("..."); break;  //正确
case 'A': printf("..."); break;  //正确，字符和整数可以相互转换
case 'A'+19: printf("..."); break;  //正确，字符和整数可以相互转换
case 9.5: printf("..."); break;  //错误，不能为小数
case a: printf("..."); break;    //错误，不能包含变量
case a+10: printf("..."); break;  //错误，不能包含变量
```

switch语句

```
switch(表达式)
{
    case 常量1: 语句1; break;
    case 常量2: 语句2; break;
    case 常量3: 语句3; break;
    ...
    case 常量n: 语句n; break;
    default: 语句n+1
}
```

如果没有与switch表达式相匹配的case常量呢?

该流程进入到default分支，执行default标号后面的语句n+1。

如果该switch结构没有default标签，此时如果没有与switch表达式相匹配的case常量，则不执行任何语句，流程转到switch结构后的语句。

switch语句 – 案例研究

```
/*switch score to grade*/
#include <stdio.h>

int main() {
    int score;
    scanf("%d",&score);
    switch (score / 10) {
        case 10:
        case 9: printf("优秀");break;
        case 8: printf("良好");break;
        case 7:
        case 6: printf("及格");break;
        default: printf("你挂科啦！");
    }
}
```

输入不同段的分数。

注意：case 10: 后面没有语句。

switch语句

Google的代码风格规范

switch 语句可以使用大括号分段，以表明 cases 之间不是连在一起的。

```
switch (var) {
  case 0: {  // 2 空格缩进
    ...      // 4 空格缩进
    break;
  }
  case 1: {
    ...
    break;
  }
  default: {
    assert(false);
  }
}
```

三目运算符

条件 ？ 表达式真 ： 表达式假

条件运算符由两个符号(?和:)组成，必须一起使用，要求有3个操作对象，称为三目运算符，是C语言唯一的一个三目运算符。

首先计算条件表达式的值是否为真，如果为真(非0)，那么求解表达式真，并且把表达式真的值作为整个条件表达式的值；如果为假(0)，那么求解表达式假，把值作为整个条件表达式的值。

注意：两个表达式必须兼容类型，否则编译错误。算术类型是兼容的，两个表达式类型不同，会发生隐式转换。

三目运算符

```
int a,b;
max = (a > b) ? a : b;
```

例如：求解两个数a、b之间的最大值

如果条件(a > b)为真，那么整个表达式的值等于a；如果为假，那么整个表达式的值等于b。

最后赋值给变量max。

三目运算符

```
int a,b;
max = (a > b) ? a : b;
```

```
int a,b;
max = a > b ? a : b;
```

1）条件运算符的优先级低于关系运算符和算术运算符，但高于赋值符。

三目运算符

```
a>b ? a : c>d ? c : d;
```

1) 条件运算符的优先级低于关系运算符和算术运算符，但高于赋值符。
2) 条件运算符的结合方向是自右至左

三目运算符

```
a>b ? a : c>d ? c : d;
```

1) 条件运算符的优先级低于关系运算符和算术运算符，但高于赋值符。
2) 条件运算符的结合方向是自右至左

```
a>b ? a : (c>d ? c : d);
```

三目运算符 – 练习

应用：三个数排序

```
/*ex-sort-three*/
#include<stdio.h>

int main() {

    int i,j,k;
    int max,min,median;

    printf("please three int number:");
    scanf("%d%d%d",&i,&j,&k);

    //请给出合适的表达式替换i，j
    max = i;
    min = j;

    median = i+j+k-max-min;

    printf("%d %d %d",max,median,min);
}
```

中山大學
SUN YAT-SEN UNIVERSITY

谢 谢

中山大学计算机学院

编制人：课题组