地球与空间科学系
DEPARTMENT OF EARTH AND SPACE SCIENCES

C程序设计基础

Introduction to C programming
Lecture 5: Decision

张振国 zhangzg@sustech.edu.cn

南方科技大学/理学院/地球与空间科学系

Review on L4

Formatted Input/Output

Bit and byte

Data types

Type casting

`scanf()`-conversion specifier

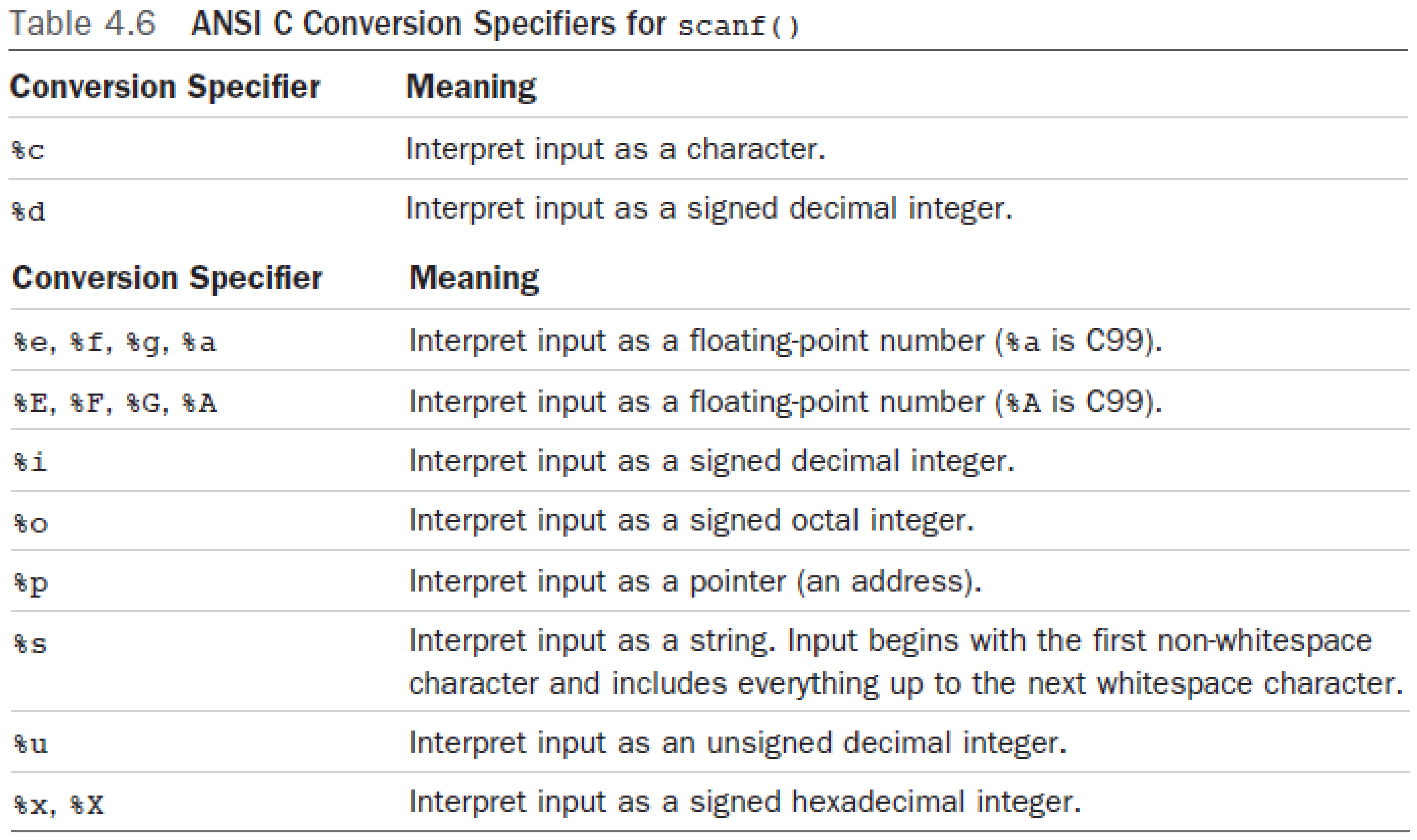


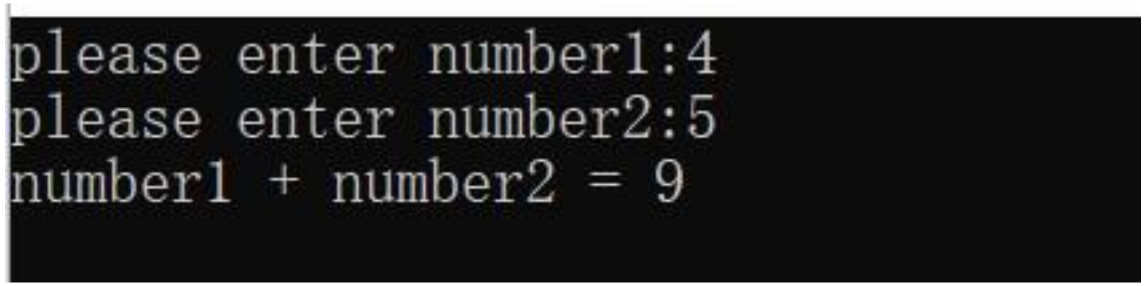
Table 4.6 **ANSI C Conversion Specifiers for `scanf()`**

Conversion Specifier	Meaning
`%c`	Interpret input as a character.
`%d`	Interpret input as a signed decimal integer.
`%e`, `%f`, `%g`, `%a`	Interpret input as a floating-point number (`%a` is C99).
`%E`, `%F`, `%G`, `%A`	Interpret input as a floating-point number (`%A` is C99).
`%i`	Interpret input as a signed decimal integer.
`%o`	Interpret input as a signed octal integer.
`%p`	Interpret input as a pointer (an address).
`%s`	Interpret input as a string. Input begins with the first non-whitespace character and includes everything up to the next whitespace character.
`%u`	Interpret input as an unsigned decimal integer.
`%x`, `%X`	Interpret input as a signed hexadecimal integer.

scanf() and printf()

Example 1: input 2 integers and make calculation

```
#include<stdio.h>
int main(void)
{ int num1;
  int num2;
  int num3=0;
  printf("please enter number1:");
  scanf("%d",&num1);
  printf("please enter number2:");
  scanf("%d",&num2);
  num3=num1+num2;
  printf("number1 + number2 = %d\n",num3);
  return 0;
}
```



```
please enter number1:4
please enter number2:5
number1 + number2 = 9
```

int num1;

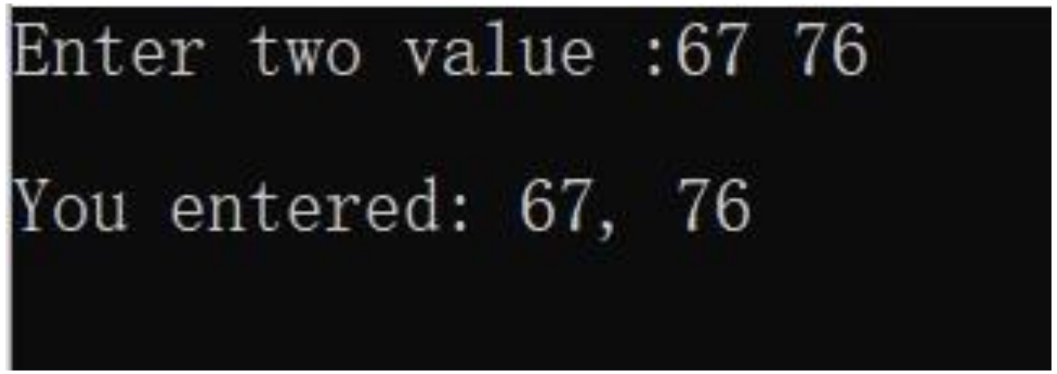
&num1

Get address of num1

scanf() and printf()

Example 2: input 2 integers in char and `int` formats

```
#include <stdio.h>
int main(void)
{
  char str[100];
  int i;

  printf( "Enter two value :");
  scanf("%s %d", str, &i);
  printf( "\nYou entered: %s, %d ", str, i);
  return 0;
}
```



```
Enter two value :67 76

You entered: 67, 76
```

int i;	**char str[100];**
&i	str

Get address of i

printf() *

```
int main(void)
{
  unsigned width, precision;
  int number = 256;
  double weight = 242.5;
  printf("Enter a field width:\n");
  scanf("%d", &width);
  printf("The number is :%*d:\n", width, number);
  printf("Now enter a width and a precision:\n");
  scanf("%d %d", &width, &precision);
  printf("Weight = %*.*f\n", width, precision, weight);
  printf("Done!\n");
  return 0;
}
```

```
Enter a field width:
6
The number is :   256:
Now enter a width and a precision:
8 3
Weight =  242.500
Done!
```

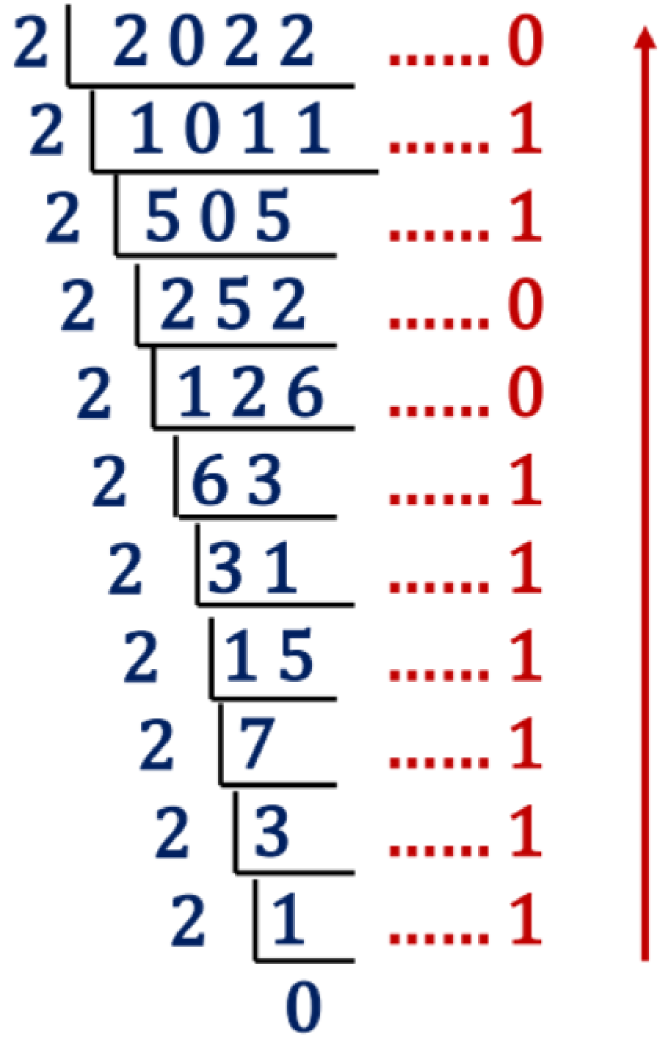
scanf() *

```c
/* skiptwo.c -- skips over first two integers of input
*/
#include <stdio.h>
int main(void)
{
  int n;
  printf("Please enter three integers:\n");
  scanf("%*d %*d %d", &n);
  printf("The last integer was %d\n", n);
  return 0;
}
```

```
Please enter three integers:
2013 2014 2015
The last integer was 2015
```

This skipping facility is useful if, for example, a program needs to read a particular column of a file that has data arranged in uniform columns.

Decimal to binary



2022

1 1 1 1 1 1 0 0 1 1 0

(float)Decimal to binary

88.8125 $\rightarrow$

$$
\begin{aligned}
& 0.8125 \\
&= 0.5 && 1\times 2^{-1} \;\ldots\ldots\; 1 \\
&+ 0.25 && 1\times 2^{-2} \;\ldots\ldots\; 1 \\
&+ 0 && 0\times 2^{-3} \;\ldots\ldots\; 0 \\
&+ 0.0625 && 1\times 2^{-4} \;\ldots\ldots\; 1
\end{aligned}
$$

$\downarrow$

88

$\downarrow$

1011000 $\searrow$

$\swarrow$ 1101

1011000.1101

Data type

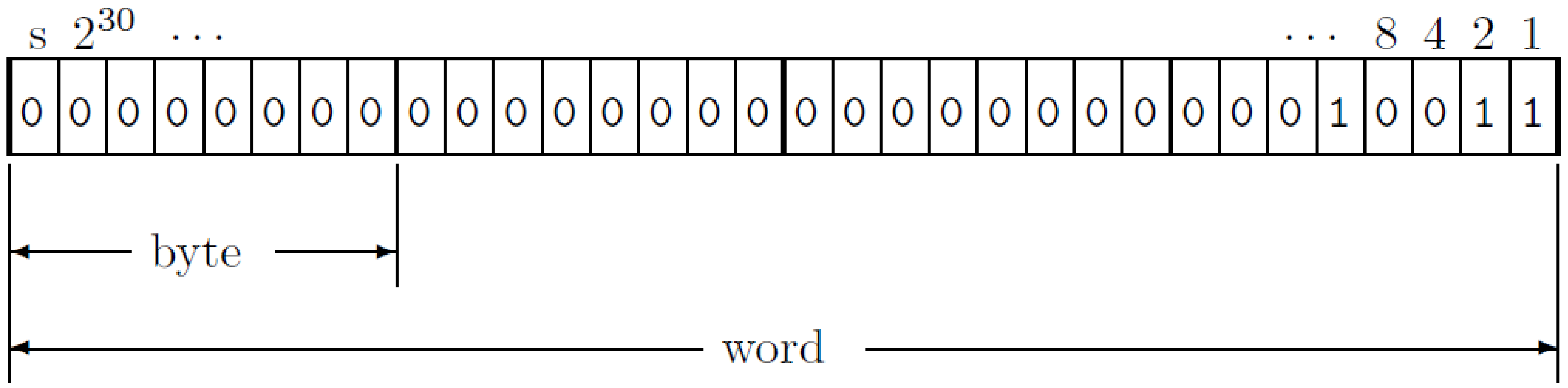
- 数据类型
  - 基本类型
    - 整型 int
    - 字符型 char
    - 实型（浮点型）
      - 单精度实型 float
      - 双精度实型 **double**
  - 构造类型
    - 数组类型
    - 结构类型 struct
    - 联合类型 union
    - 枚举类型 enum
  - 指针类型
  - 空类型（无值类型） void

Integer

有符号基本整型	(signed)int
有符号短整型	(signed)short (int)
有符号长整型	(signed) long (int)
无符号基本整型	unsigned int
无符号短整型	unsigned short (int)
无符号长整型	unsigned long (int)

( )表示默认情况

Use byte to store integer number



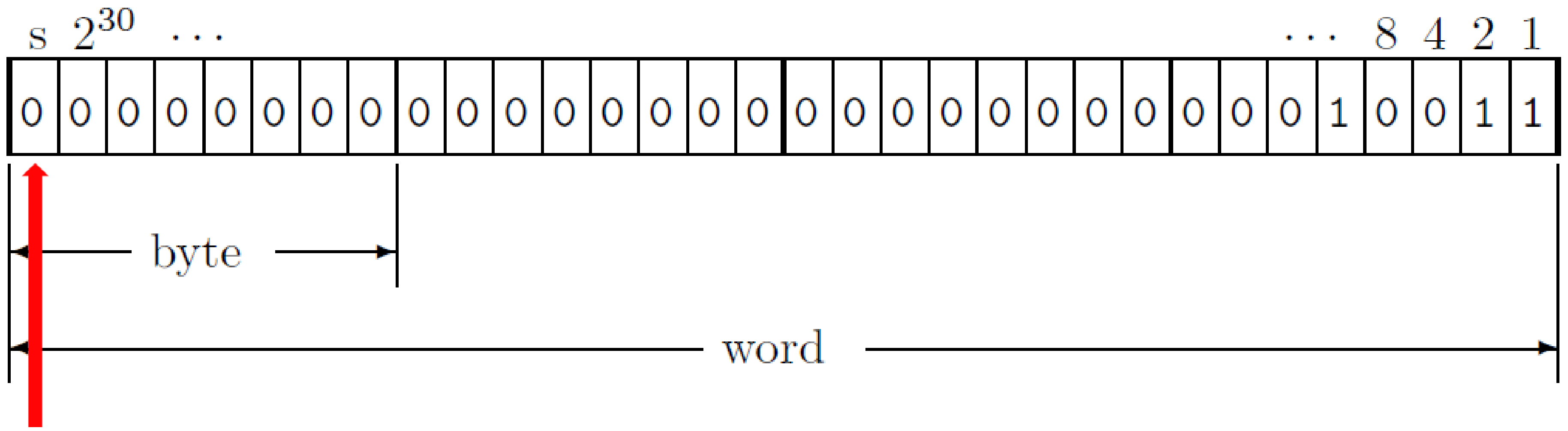
for an `int` type
sizeof(int) = 4

$$A = \boxed{\sum_{i=1}^{32} 2^{i-1} * r_i} =$$

Positive numbers:	N
Negative numbers:	$2^{32} + N$

$$A = \begin{cases} A & A \le 2^{31} - 1 \\ A - 2^{32} & A \ge 2^{31} \end{cases}$$

Use byte to store integer number



for an `int` type
sizeof(int) = 4

Sign bit:
 0 for positive
 1 for negative

$$A = \boxed{\sum_{i=1}^{32} 2^{i-1} * r_i} =$$

Positive numbers:	N
Negative numbers:	$2^{32} + N$

$$A = \begin{cases} A & A \leq 2^{31} - 1 \\ A - 2^{32} & A \geq 2^{31} \end{cases}$$

Integer

类型	类型说明符	长度	数的范围
基本型	int	4字节（大部分）	-2^{31}～$2^{31}-1$
短整型	short	2字节	-2^{15}～$2^{15}-1$
长整型	long	4字节	-2^{31}～$2^{31}-1$
双长型	long long	8字节	-2^{63}～$2^{63}-1$
无符号整型	unsigned	4字节	0～（$2^{32}-1$）
无符号短整型	unsigned short	2字节	0～65535
无符号长整型	unsigned long	4字节	0～（$2^{32}-1$）

Integer

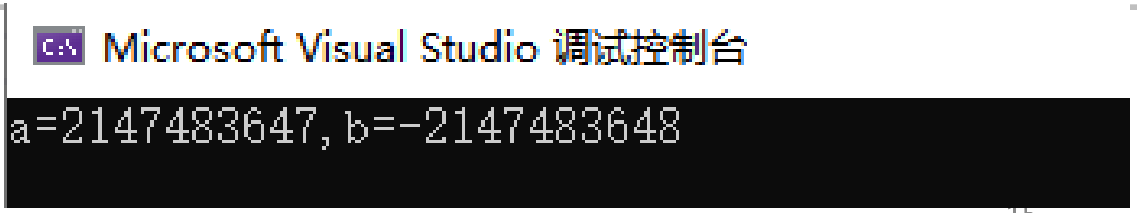
- Larger than the largest positive value?

```
 01111111111111111111111111111111
                               +1
 ________________________________
 10000000000000000000000000000000  =  2^31
```

$2^{32}+N=2^{31}$,

$\Rightarrow N=-2^{31}$

```
#include <stdio.h>
void  main()
{int  a,b;
 a=2147483647;
 b=a+1;
 printf( “%d,%d\n” ,a,b);
}
```



```
a=2147483647,b=-2147483648
```

Floating-Point Types

浮点型常量的表示方法：

两种表示形式 $\begin{cases} \text{小数decimal} & 0.123 \\ \text{指数exponent} & 3e-3 \end{cases}$

注意:字母e(或E)之前必须有数字，且e后面的指数必须为整数

√ 1e3、1.8e-3、-123e-6、-.1e-3

× e3、2.1e3.5、.e3、e

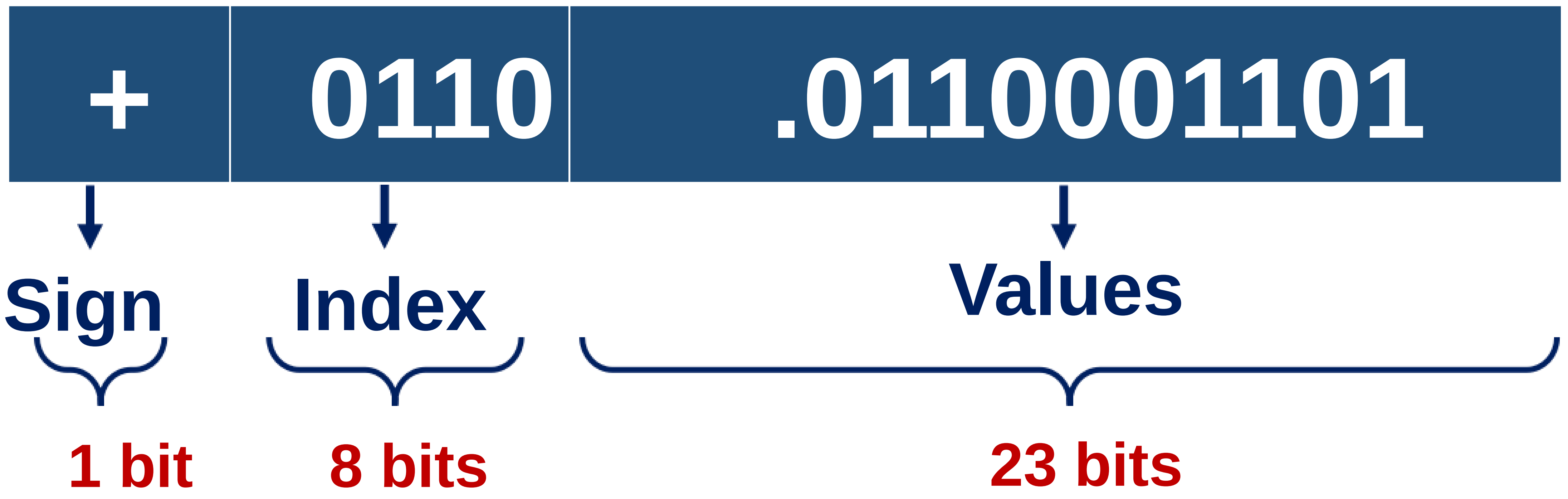
Use byte to store real number

Real number = rational number (10, -0.23) + irrational number (PI, $\sqrt{2}$)

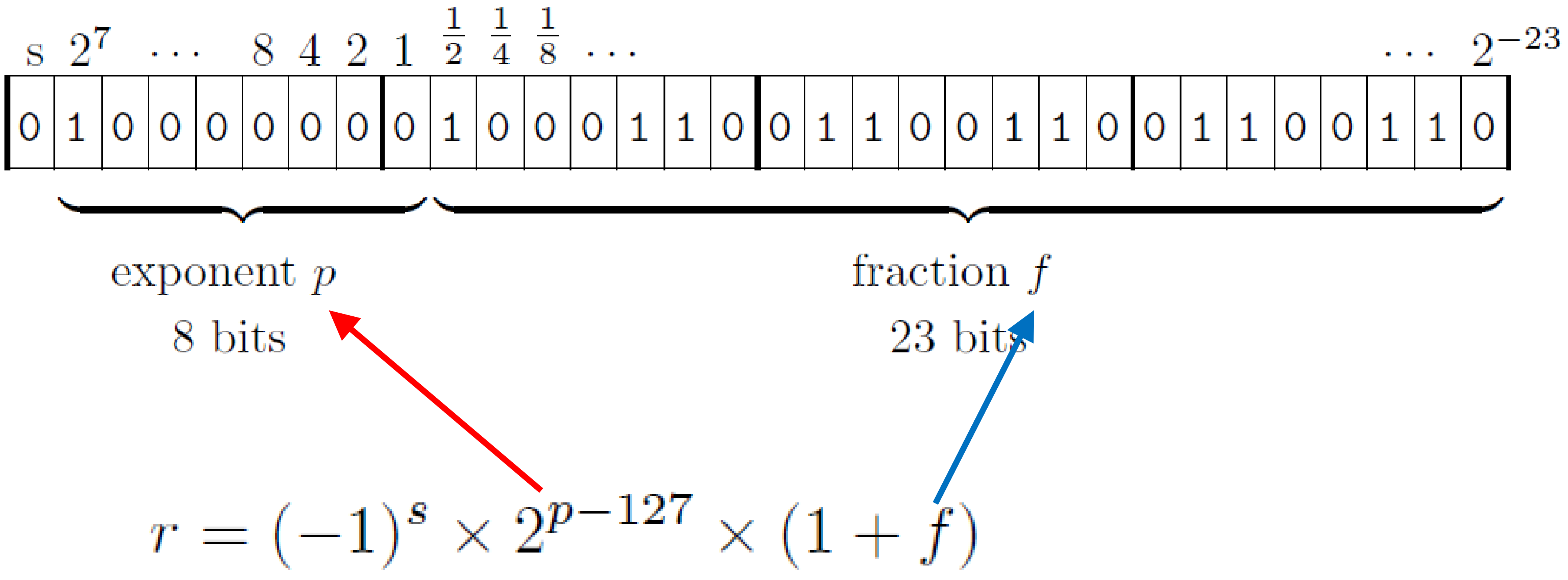
Use 2 as basis

How to use byte to denote 88.8125?!

$$88.8125 = 1011000.1101 = 1.0110001101 \times 2^{6}$$



Floating-Point Types



$$r = (-1)^s \times 2^{p-127} \times (1+f)$$

$\pm$mantissa*2^{exponent}

假数(对数之小数部份)

Floating-Point Types

浮点型变量分为单精度（float型）、双精度（double型）和长双精度型（long double）三类形式。 $r = (-1)^s \times 2^{p-127} \times (1+f)$

Types	Bits	Exponent bias	Range	Precision digits
float	32(8+23)	127	$10^{-38} \sim 10^{38}$	6-8
double	64(11+52)	1023	$10^{-308} \sim 10^{308}$	15-17
long double	128(15+112)	16383	$10^{-4931} \sim 10^{4932}$	33-34

It depends on the machine

bits = sign bit + (exponent + fraction) bits

float: round-off error

```c
#include<stdio.h>
int main(void)
{ float a1, a2, b, c;
  a1 = 123456789.2f;

  b = a1 + 500;
  c = b - 500;
  a2  = c;

  if(a1 == a2)
  {printf("a1 = a2. \n");}
  else
  {printf("a1 /= a2. \n");}
  printf("a1=%f\n", a1);
  printf("a2=%f\n", a2);
  printf("b=%f\n", b);
  printf("c=%f\n", c);
  return 0;
}
```

舍入误差是指运算得到的近似值和精确值之间的差异。比如当用有限位数的浮点数来表示实数的时候(理论上存在无限位数的浮点数)就会产生舍入误差。舍入误差是量化误差的一种形式。 如果在一系列运算中的一步或者几步产生了舍入误差，在某些情况下，误差会随着运算次数增加而积累得很大，最终得出没有意义的运算结果。

```
a1 /= a2.
a1=123456792.000000
a2=123456800.000000
b=123457296.000000
c=123456800.000000
```

```c
a1 = 1234.2f;
```

```
a1 = a2.
a1=1234.199951
a2=1234.199951
b=1734.199951
c=1234.199951
```

Suppl.Round-off errors

- During the Gulf War in 1991, a U.S. Patriot missile failed to intercept an Iraqi Scud missile, and 28 Americans were killed.
- A later study determined that the problem was caused by the inaccuracy of the binary representation of 0.10.

— The Patriot incremented a counter once every 0.10 seconds.

— It multiplied the counter value by 0.10 to compute the actual time.

- However, the (24-bit) binary representation of 0.10 actually corresponds to 0.099999904632568359375, which is off by 0.000000095367431640625.
- This doesn’t seem like much, but after 100 hours the time ends up being off by 0.34 seconds — enough time for a Scud to travel 500 meters!
- Professor Skeel wrote a short article about this.
- Roundoff Error and the Patriot Missile. SIAM News, 25(4):11, July 1992.

Suppl.Floating-point

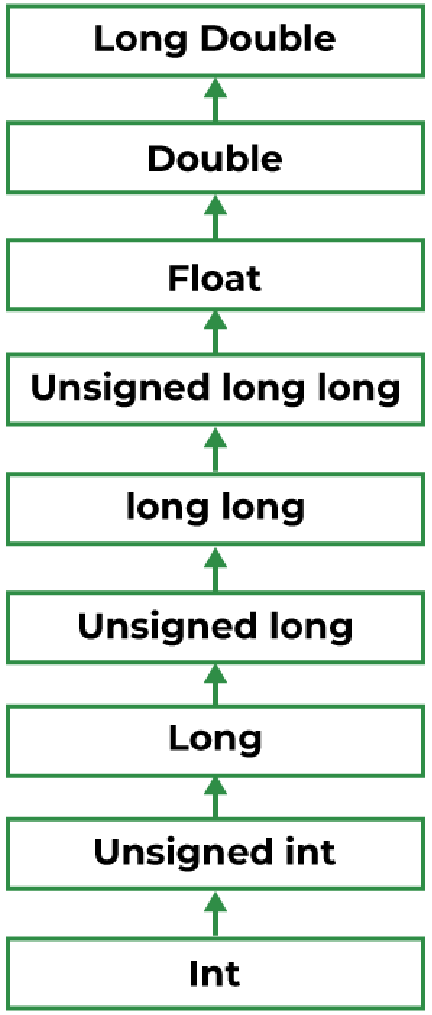
- With 32 bits, there are 2^{32}, or about 4 billion, different bit patterns.
  - These can represent 4 billion integers or 4 billion reals.
  - But there are an infinite number of reals, and the IEEE format can only represent some of the ones from about -2^{128} to $+2^{128}$.
  - Represent same number of values between 2^n and 2^{n+1} as 2^{n+1} and 2^{n+2}
- Thus, floating-point arithmetic has “issues”
  - Small round-off errors can accumulate with multiplications or exponentiations, resulting in big errors.
  - Rounding errors can invalidate many basic arithmetic principles such as the associative law, $(x + y) + z = x + (y + z)$.
- The IEEE 754 standard guarantees that all machines will produce the same results—but those results may not be mathematically accurate!

Type casting: Implicit Type Casting I

- Implicit type casting in C is used to convert the data type of any variable without using the actual value that the variable holds. It performs the conversions without altering any of the values which are stored in the data variable. Conversion of lower data type to higher data type will occur automatically.

- Integer promotion will be performed first by the compiler. After that, it will determine whether two of the operands have different data types. Using the hierarchy below, the conversion would appear as follows if they both have varied data types:

```
1.0f/2;
sqrt(2.0+1);
```



Type casting: Implicit Type Casting II

```
// C program to illustrate the use of
//typecasting
#include <stdio.h>
int main()
{       int a = 15, b = 2;
        float div;
        div = a / b;
        printf("The result is %f\n", div);
        return 0;
}
```

Output:

```
The result is 7.000000
```

Homework4 Feedback

1. Write a program that reads in a floating-point number and prints it first in decimal-point notation, then in exponential notation. Have the output use the following format (the actual number of digits displayed for the exponent depends on the system):

```
Enter a floating-point value: 64.25
fixed-point notation: 64.250000
exponential notation: 6.425000e+01
```

printf("%e", num)

3.0e-23

3. The mass of a single molecule of water is about 3.0×10^{-23} grams. A quart of water is about 950 grams. Write a program that requests an amount of water, in quarts, and displays the number of water molecules in that amount.

MB(Byte) & Mb(bit): 1 Byte = 8 bit

4. Write a program that requests the download speed in megabits per second (Mbs) and the size of a file in megabytes (MB). The program should calculate the download time for the file. Note that in this context one byte is eight bits. Use type float, and use / for division. The program should report all three values (download speed, file size, and download time) showing two digits to the right of the decimal point, as in the following:

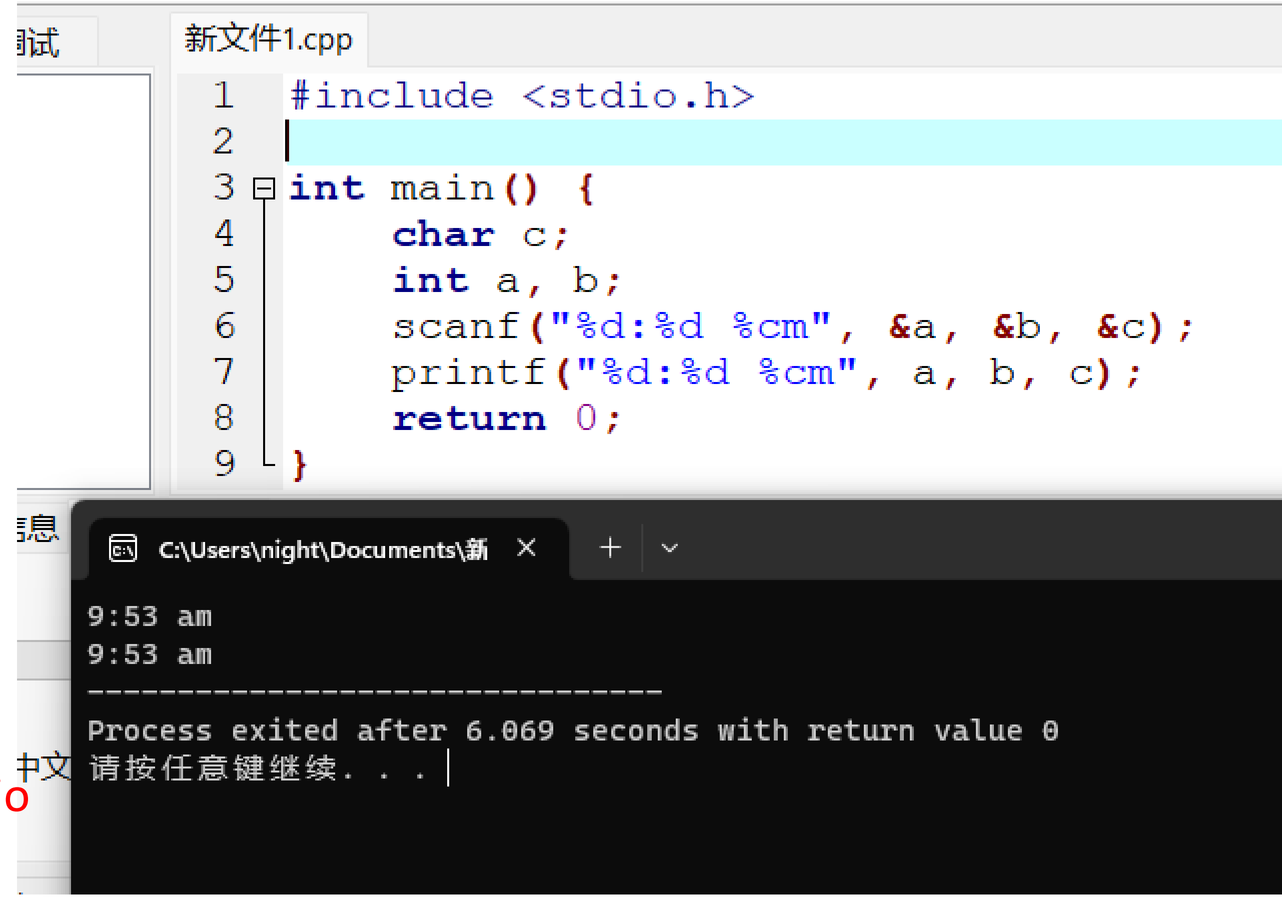
At 18.12 megabits per second, a file of 2.20 megabytes downloads in 0.97 seconds.

6. Write a program that asks the user for a 12-hour time, then displays the time in 24-hour form:

Enter a 12-hour time: 9:11 PM

Equivalent 24-hour time : 21:11

(Tips) The format of the input can be controlled by the scanf statement.


新文件1.cpp

```c
#include <stdio.h>

int main() {
	char c;
	int a, b;
	scanf("%d:%d %cm", &a, &b, &c);
	printf("%d:%d %cm", a, b, c);
	return 0;
}
```

C:\Users\night\Documents\新

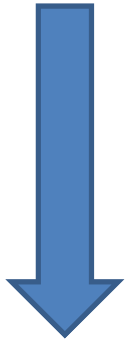
```
9:53 am
9:53 am
--------------------------------
Process exited after 6.069 seconds with return value 0
请按任意键继续. . .
```

scanf(“%d:%d %cm” , &a, &b, &c); // Dev-cpp
scanf_s("%d:%d %cm", &a, &b, &c, 10); // Visual Studio

Objective of this lecture

You can use C to control the workflow!

Straight-line code



Flow of control

Content

1. **Relational and Logical operators**
2. **If statement**
3. **Switch statement**

"To be, or not to be, that is the question"

沙士比亚-哈姆雷特

Relational Operators

More examples on different data types

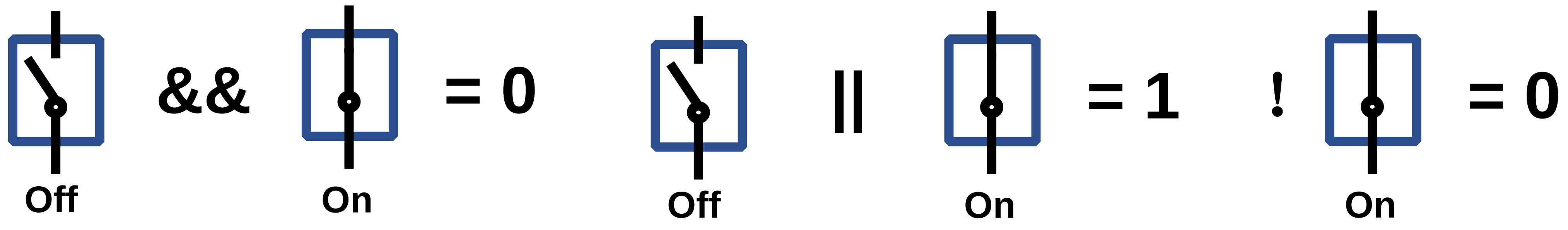
Operators	**float A = 3.5, B = 3.5;**	**char A = 'A', B = 'B';**
==	A==B = 1 (true)	A==B = 0 (false)
!=	A != B = 0 (false)	A != B = 1 (true)
>	A > B = 0 (false)	A > B = 0 (false)
<	A < B = 0 (false)	A < B = 1 (true)
>=	A >= B = 1 (true)	A >= B = 0 (false)
<=	A <= B = 1 (true)	A <= B = 0 (true)

Logical/boolean operators

Define two variables: int A = 0, B = 1;

Operators	Description	Example
&&	AND operator, if both are on, then on	A&&B = 0 (false)
||	OR operator, if any is on, then on	A ||B = 1 (true)
!	NOT operator, turn opposite	!A = 1 (true)!B = 0 (false)

Logical/boolean operators



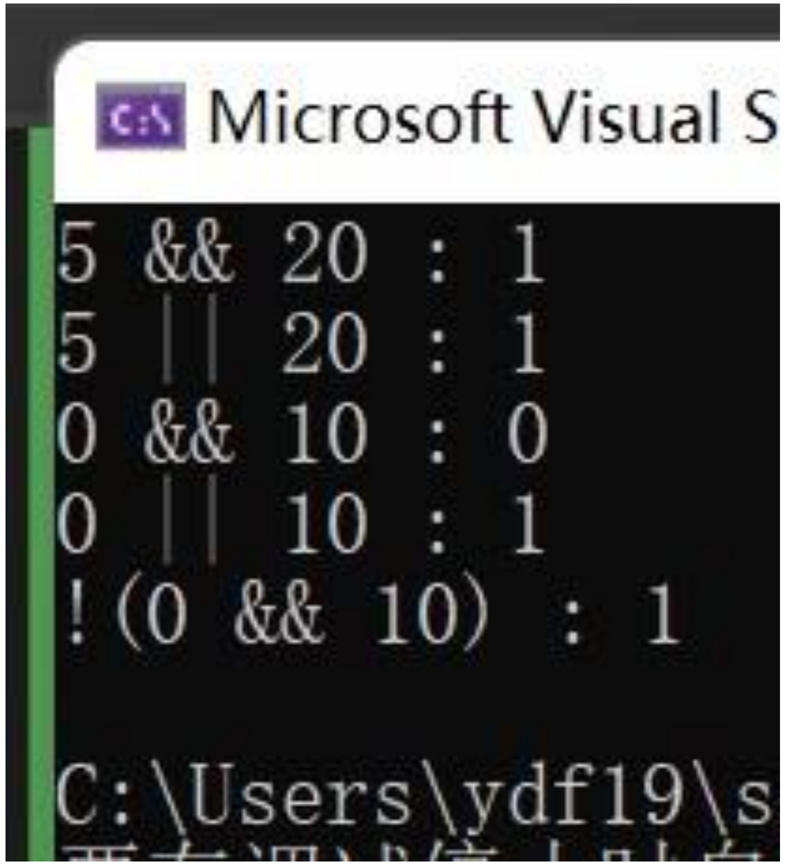
Examples:

`6 > 2 && 3 == 3`	True.
`! (6 > 2 && 3 == 3)`	False.
`x != 0 && (20 / x) < 5`	The second expression is evaluated only if `x` is nonzero.

Logical/boolean operators

Example : comparing integers

```
#include <stdio.h>
main()
{
    int a = 5;
    int b = 20;
    int c;
    c = a && b;
    printf("5 && 20 : %d\n",c);
    c = a || b;
    printf("5 || 20 : %d\n",c);
    a = 0;
    b = 10;
    c = a && b;
    printf("0 && 10 : %d\n", c);
    c = a || b;
    printf("0 || 10 : %d\n", c);
    c = !(a && b);
    printf("!(0 && 10) : %d\n", c);
}
```



```
5 && 20 : 1
5 || 20 : 1
0 && 10 : 0
0 || 10 : 1
!(0 && 10) : 1

C:\Users\ydf19\s
```

Logical/boolean operators

short-circuit evaluation “短路” 计算

```
&&

||
```

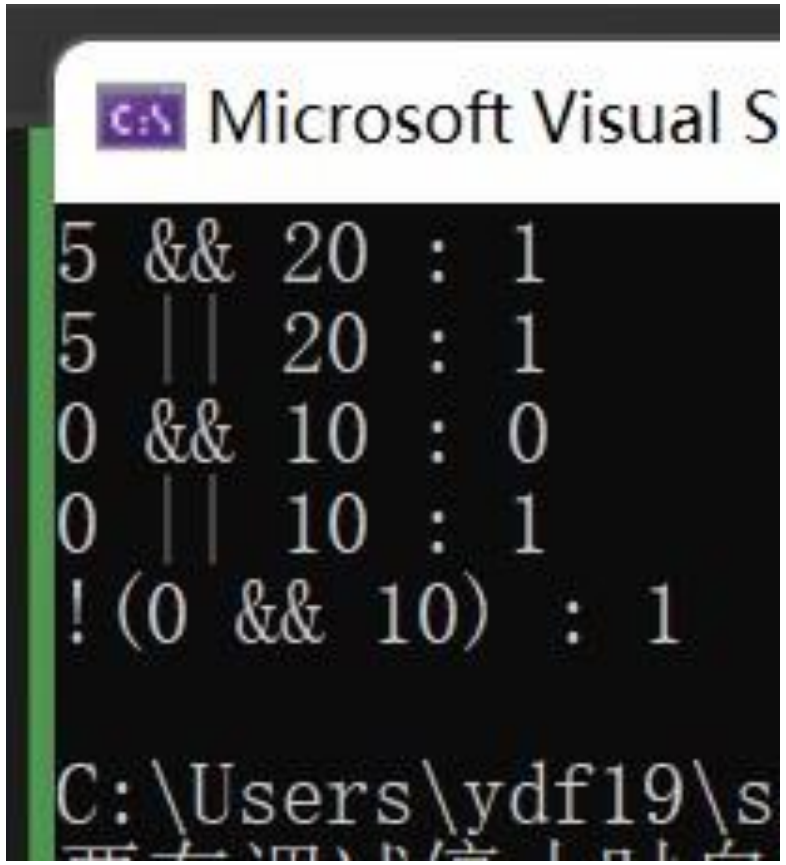
These operators first evaluate the left operand, then the right operand.
If the value of the expression can be deduced from the value of the left operand alone, then the right operand isn’t evaluated.

```
(i!=0)&&(j/i>0)
```

```
i>0 && ++j>0
```

此种情况会影响结果

Logical/boolean operators



```
5 && 20 : 1
5 || 20 : 1
0 && 10 : 0
0 || 10 : 1
!(0 && 10) : 1

C:\Users\ydf19\s
```

- 早期C没有logical 变量，用int数据指示
  - ✓ 0 for false
  - ✓ others for true
- C99新增_Bool
  - ✓ _Bool a = 1;

Logical/boolean operators

```
#include<stdio.h>
#include<stdbool.h>
int main(void){
   _Bool flag;
   bool x, y;
   flag = 5;
   printf("flag= %d\n", flag);
   x = false;
   y = true;
   printf("false= %d\n", x);
   printf("true= %d\n", y);
   return 0;
}
```

- C99新增_Bool
- C99新增<stdbool.h>

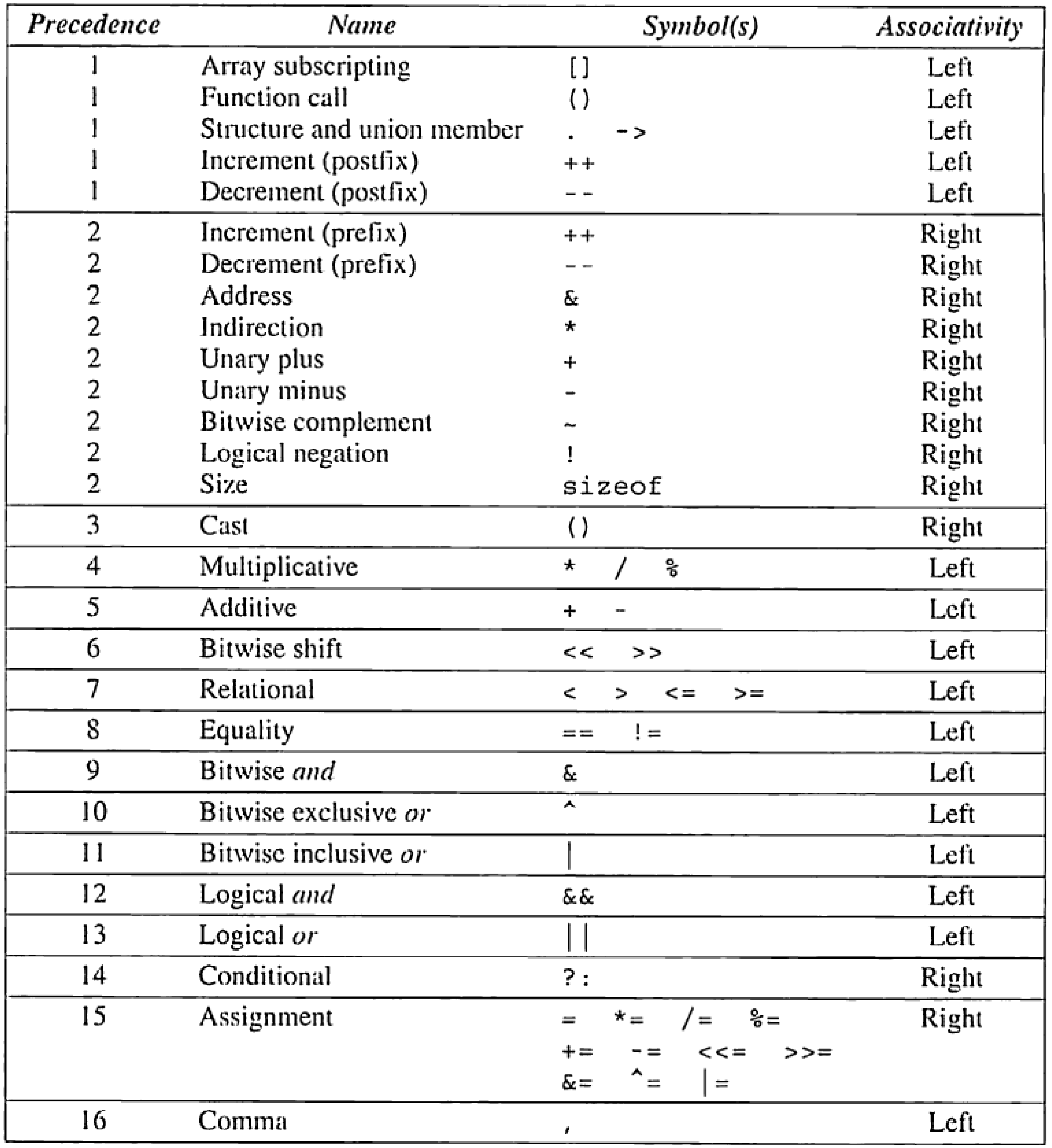
```
flag= 1
false= 0
true= 1
```

Precedence

a<=b && b<=c

```
include<stdbool.h>
flat a=2.5,b=7.5,c=5.0,d=6.0;
printf(“%d”,c/2.0+d <a && !true||c<=d
```

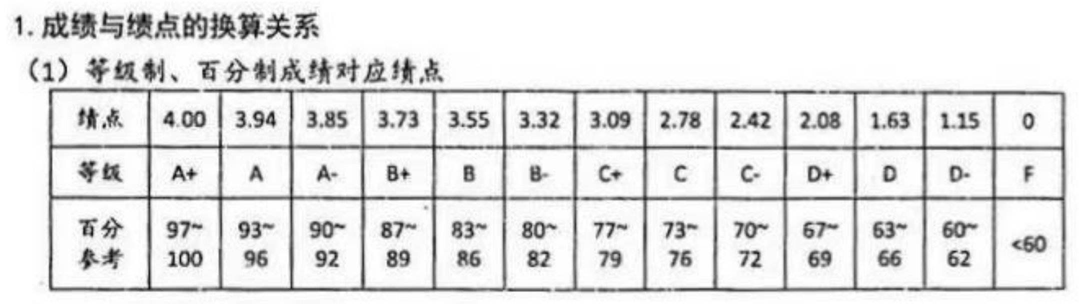
1



Precedence	*Name*	*Symbol(s)*	*Associativity*
1	Array subscripting	`[]`	Left
1	Function call	`()`	Left
1	Structure and union member	`.` `->`	Left
1	Increment (postfix)	`++`	Left
1	Decrement (postfix)	`--`	Left
2	Increment (prefix)	`++`	Right
2	Decrement (prefix)	`--`	Right
2	Address	`&`	Right
2	Indirection	`*`	Right
2	Unary plus	`+`	Right
2	Unary minus	`-`	Right
2	Bitwise complement	`~`	Right
2	Logical negation	`!`	Right
2	Size	`sizeof`	Right
3	Cast	`()`	Right
4	Multiplicative	`*` `/` `%`	Left
5	Additive	`+` `-`	Left
6	Bitwise shift	`<<` `>>`	Left
7	Relational	`<` `>` `<=` `>=`	Left
8	Equality	`==` `!=`	Left
9	Bitwise *and*	`&`	Left
10	Bitwise exclusive *or*	`^`	Left
11	Bitwise inclusive *or*	`|`	Left
12	Logical *and*	`&&`	Left
13	Logical *or*	`||`	Left
14	Conditional	`?:`	Right
15	Assignment	`=` `*=` `/=` `%=` `+=` `-=` `<<=` `>>=` `&=` `^=` `|=`	Right
16	Comma	`,`	Left

Decision-making in life

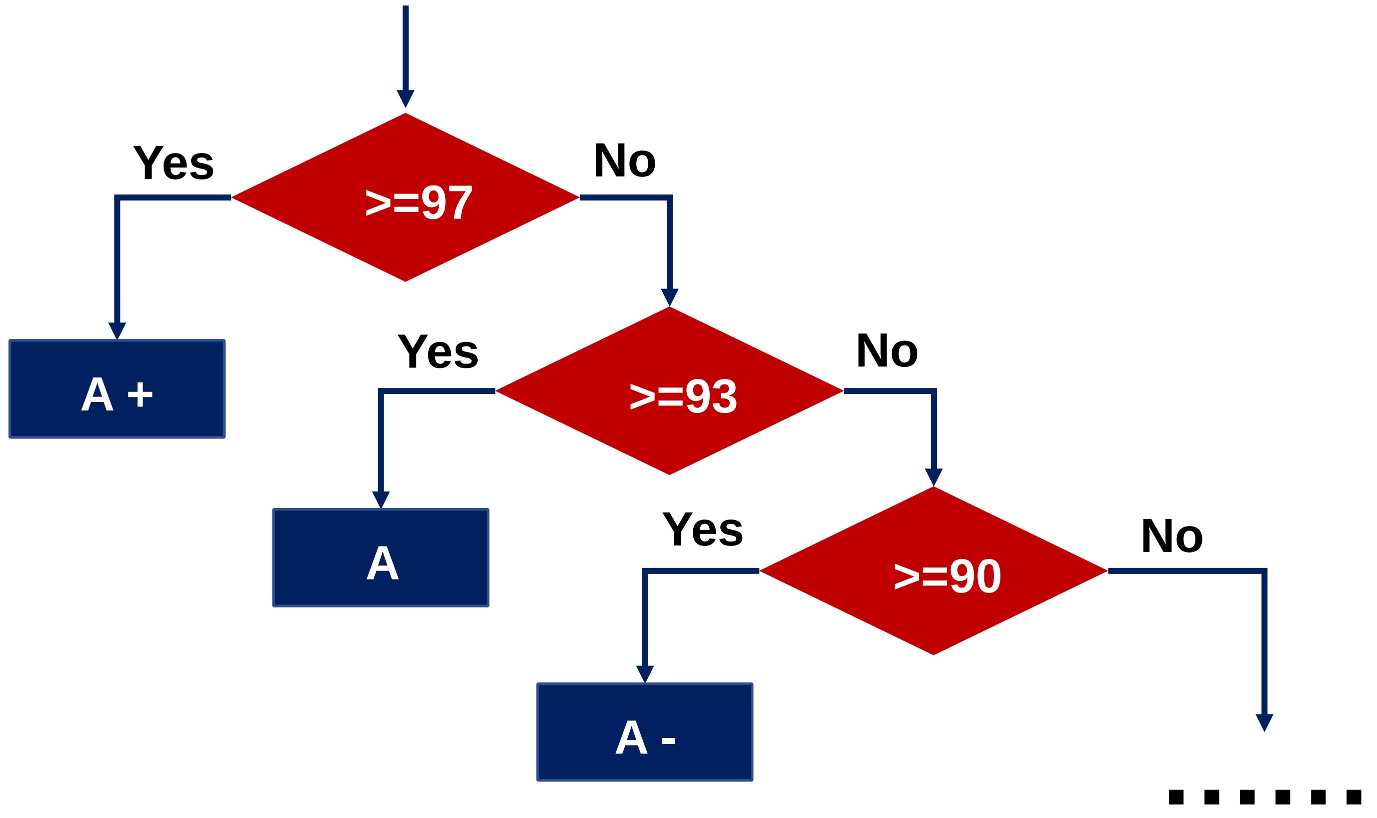
Decision-making in life



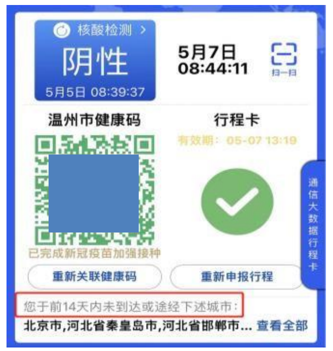
1. 成绩与绩点的换算关系

（1）等级制、百分制成绩对应绩点

绩点	4.00	3.94	3.85	3.73	3.55	3.32	3.09	2.78	2.42	2.08	1.63	1.15	0
等级	A+	A	A-	B+	B	B-	C+	C	C-	D+	D	D-	F
百分参考	97~100	93~96	90~92	87~89	83~86	80~82	77~79	73~76	70~72	67~69	63~66	60~62	<60



Decision-making in life



核酸检测 >
阴性
5月5日 08:39:37

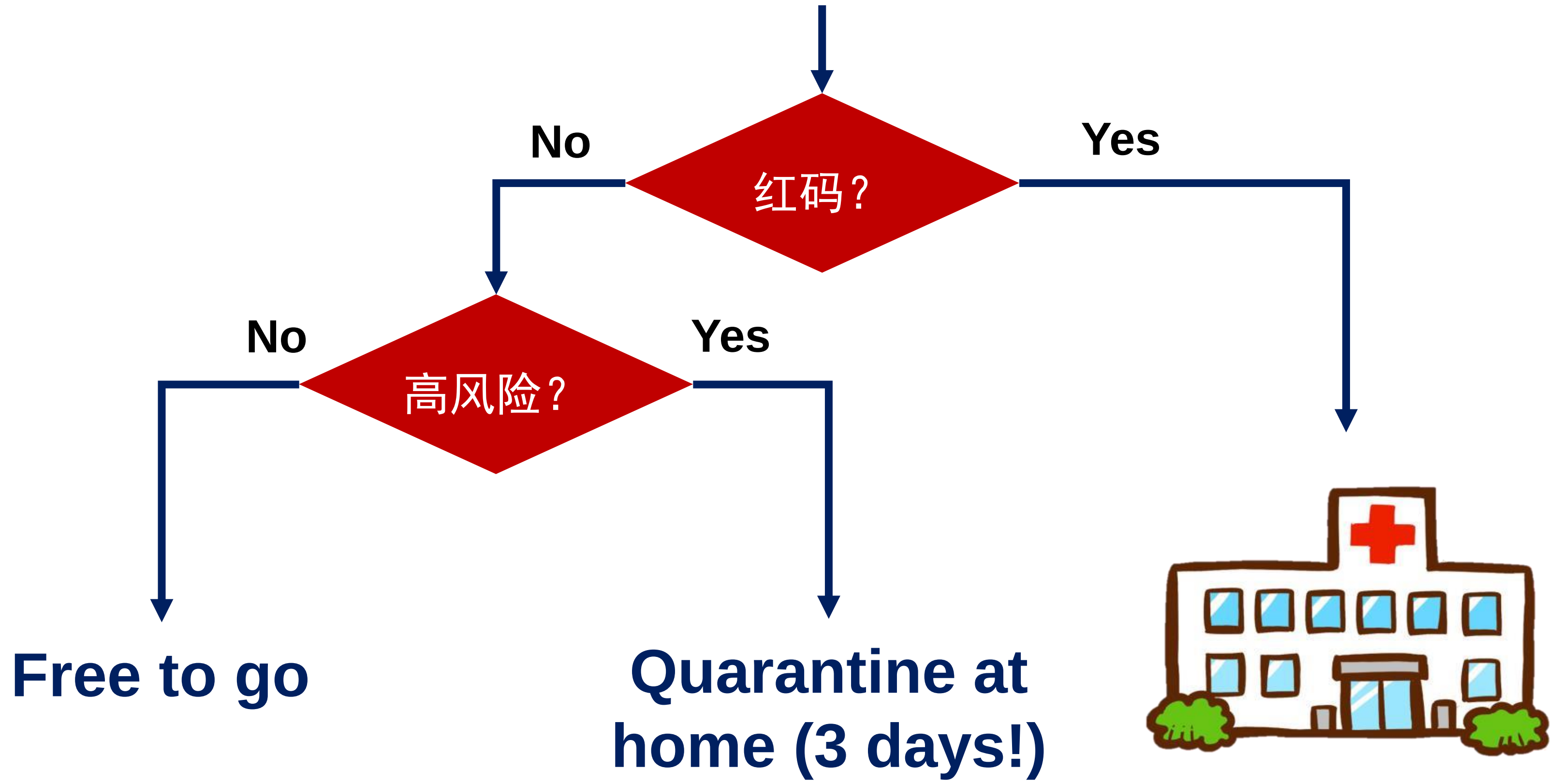
5月7日
08:44:11
扫一扫

温州市健康码
[redacted]
已完成新冠疫苗加强接种
重新关联健康码

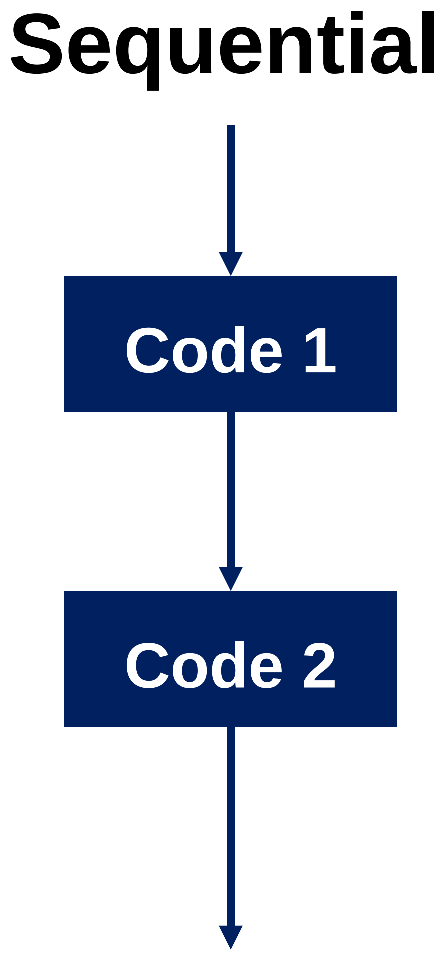
行程卡
有效期：05-07 13:19
重新申报行程

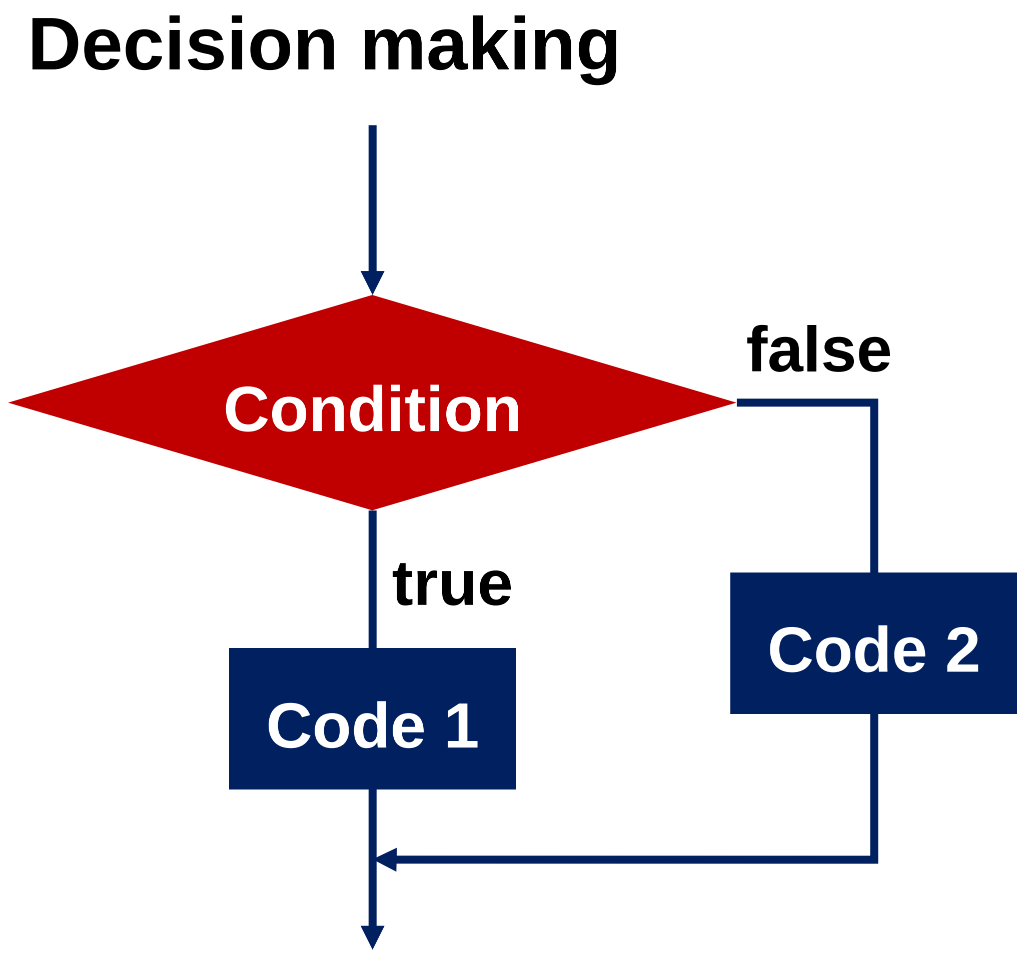
通信大数据行程卡

您于前14天内未到达或途经下述城市：
北京市,河北省秦皇岛市,河北省邯郸市... 查看全部



Decision-making in life





Content

If statement

If statement has a boolean expression followed by one or more statements.

```
if(boolean_expression)
{ /* code 1 */ }
```

```
if(boolean_expression)
{ /* code 1 */ }
else
{ /* code 2 */ }
```

If statement

if(condition)	**如果**(条件满足)
{option A}	{A选项}
else	**否则**
{option B}	{B选项}

If statement

condition

Ture/Yes/1

(2<5, 3==3, ’a’!=‘b’)

False/No/0

**(1>=5, 3<1,
’a’==‘b’)**

If and if-else

What does condition mean?

```
int a = 3;

if (a > 10)
{
   //...
}else
{
   //...
}
```

```
float f = 10;

if (a == 10)
{
   //...
}else
{
   //...
}
```

```
char c = ‘A’;

if (a != ‘A’)
{
   //...
}else
{
   //...
}
```

If and if-else

If only

```
int a = 3;

if (a > 10)
{
   printf(“a>10”);
}

return 0;
```

Block 1

If else

```
int a = 3;

if (a > 10)
{
   printf(“a>10”);
}else{
   printf(“a<10”);
}

return 0;
```

Block 1

Block 2

If and if-else

```
if (a > 10)
{
   printf(“a>10”);
}
```

```
if (a > 10)
   printf(“a>10”);
```

```
if (a > 10) printf(“a>10”);
```

```
if (a > 10)
{
   printf(“a>10”);
}
else
{
   printf(“a<10”);
}
```

```
if (a > 10)
   printf(“a>10”);
else
   printf(“a<10”);
```

```
if (a > 10) printf(“a>10”);
else printf(“a<10”);
```

If and if-else

Question?

```
int a = 5;
if (a > 10);
   printf(“a>10”);
```

```
int a = 15;
if (a > 10);
   printf(“a>10”);
```

a>10

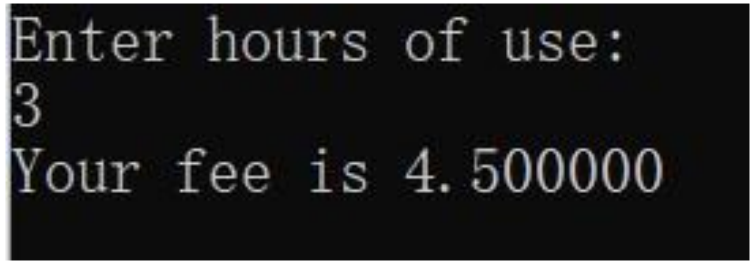
```
if (a > 10);
   printf(“a>10”);
else
   printf(“a<10”);
```

error

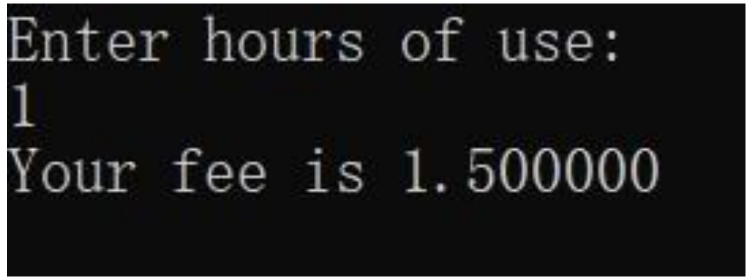
Case study: If statement

Case: calculate the shared bike fee (<1h is 1h)!

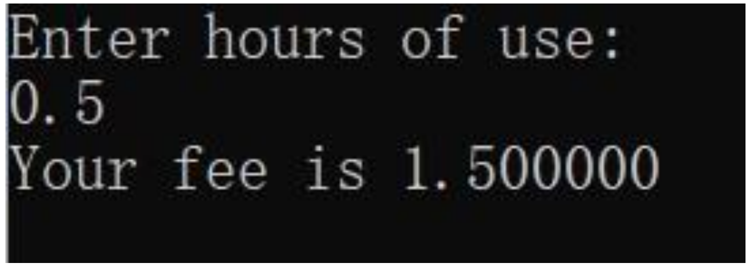
```
#include <stdio.h>
main ()
{
    float hours, fee;
    printf(“Enter hours of use:\n");
    scanf("%f", &hours);
    if(hours < 1)
    {
        hours = 1;
    }
    fee = 1.5 * hours;
    printf("Your fee is %f", fee); }
```



```
Enter hours of use:
3
Your fee is 4.500000
```



```
Enter hours of use:
1
Your fee is 1.500000
```

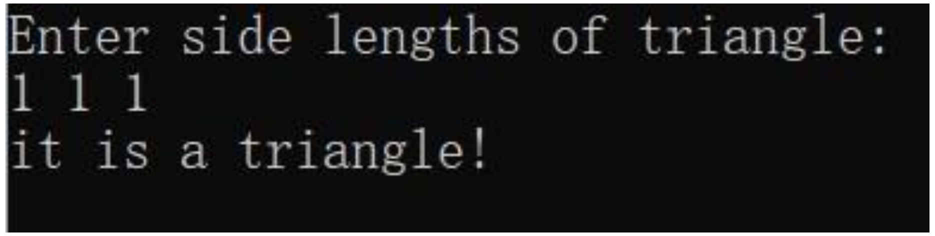


```
Enter hours of use:
0.5
Your fee is 1.500000
```

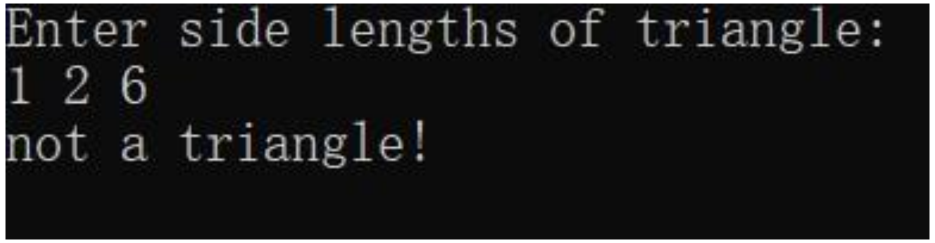
Case study: If statement

Case: check if three sides can form a triangle

```
#include <stdio.h>
main()
{
    float a,b,c;
    printf("Enter side lengths of triangle:\n");
    scanf("%f %f %f", &a, &b, &c);
    if(a+b>c && a+c>b && b+c>a)
    {
    printf("it is a triangle!");
    }else
    {
    printf("not a triangle!");
    }
}
```



```
Enter side lengths of triangle:
1 1 1
it is a triangle!
```



```
Enter side lengths of triangle:
1 2 6
not a triangle!
```

If - else if

If-elseif has more boolean expression followed by more statements.

```
if( condition 1 )
{ /* code 1 */ }
elseif( condition 2 )
{ /* code 2 */ }
elseif( condition 3 )
{ /* code 3 */ }
elseif( condition 4 )
{ /* code 4 */ }
…
else
{/* code N */}
```

If - else if

Block 1

```
int a = 3;

if (a > 10)
{
    //...
}
```

Block 1

Block 2

```
int a = 3;

if (a > 10)
{
    //...
}else{
    //...
}
```

Block 1

Block 2

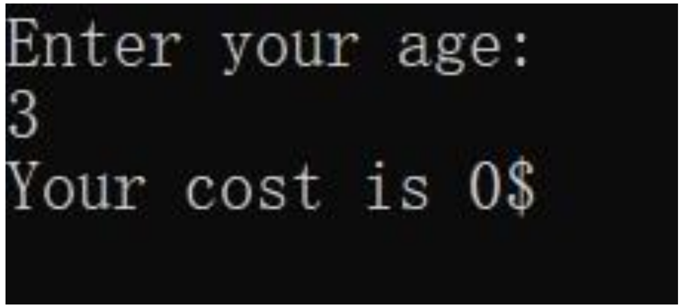
Block 3

```
int a = 3;

if (a == 10)
{
    //...
}elseif(a == 2)
{
    //...
}elseif(a == 3)
{
    //...
}
```

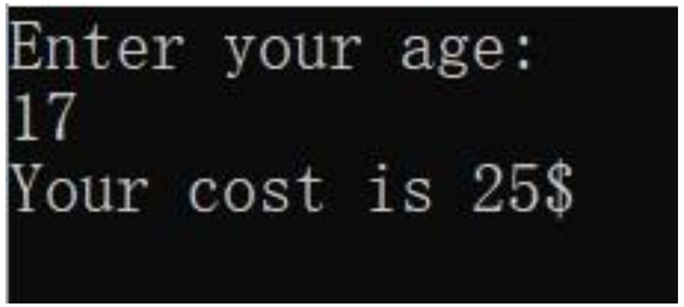
Case study: If - else if

Case: what is the cost of attendance?

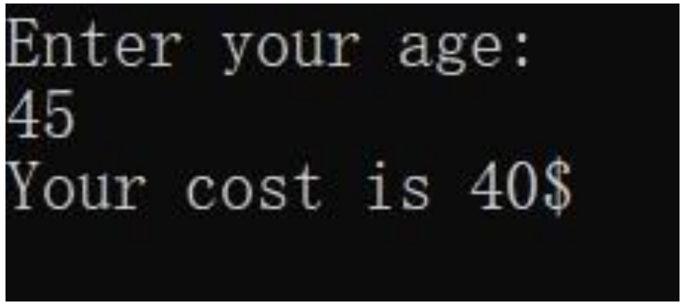
```
#include <stdio.h>
main()
{
    int a;
    printf("Enter your age:\n");
    scanf("%d", &a);
    if( a < 10 )
    {
        printf("Your cost is 0$\n" );
    }
    else if( a >= 10 && a < 20 )
    {
        printf("Your cost is 25$\n" );
    }
    else
    {
        printf("Your cost is 40$\n" );
    }
}
```



```
Enter your age:
3
Your cost is 0$
```



```
Enter your age:
17
Your cost is 25$
```

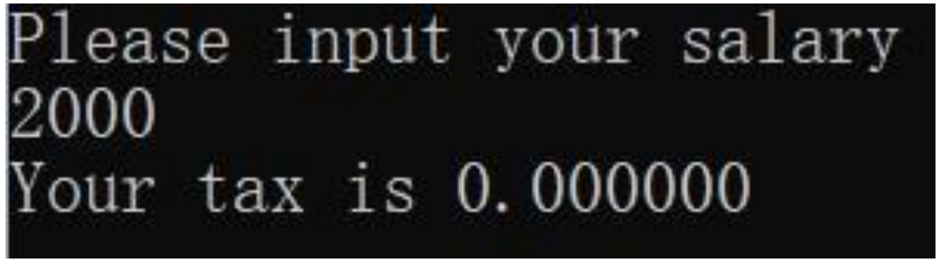


```
Enter your age:
45
Your cost is 40$
```

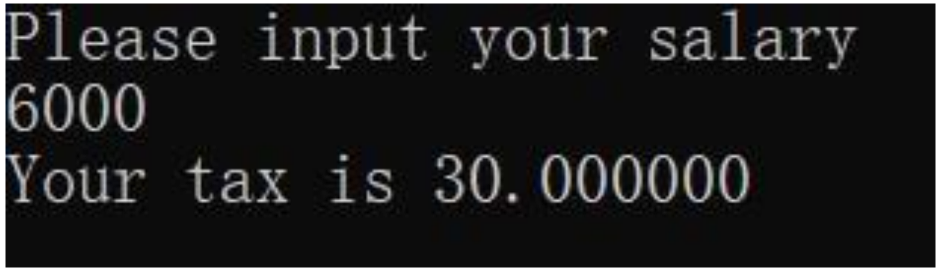
Case study: If - else if

Case: calculate the tax based on salary

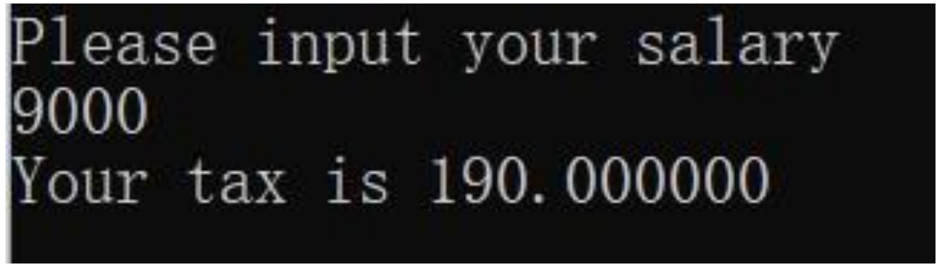
```
#include <stdio.h>
main()
{
    double salary, tax;
    printf("Please input your salary\n");
    scanf("%lf", &salary);
    if(salary <= 5000) {
        tax = 0;
    }
    else if(salary <= 8000) {
        tax = (salary - 5000) * 0.03;
    }
    else{
        tax = 90 + (salary - 8000) * 0.1;
    }
    printf("Your tax is %lf\n", tax);
}
```



```
Please input your salary
2000
Your tax is 0.000000
```



```
Please input your salary
6000
Your tax is 30.000000
```



```
Please input your salary
9000
Your tax is 190.000000
```

？ statement

expression： **expression1 ? expression2 : expression3**

- If *expression1* is true (nonzero), the whole conditional expression has the same value as *expression2*. If *expression1* is false (zero), the whole conditional expression has the same value as *expression3* .

```
if (y < 0)
      x = -y;
else
     x = y;
```

$\longrightarrow$

```
x = (y < 0) ? -y : y;
```

？ statement

expression： expression1 ? expression2 : expression3

- 条件运算符的执行顺序：先求解表达式 1 ，若为非 0 （真）则求解表达式 2 ，此时表达式 2 的值就作为整个条件表达式的值。若表达式 1 的值为 0 （假），则求解表达式 3 ，表达式 3 的值就是整个条件表达式的值。
- 条件运算符优先级高于赋值运算符 ，低于关系运算符和算术运算符。
- 条件运算符的结合方向为“自右至左”。
- “表达式2”和“表达式3”不仅可以是数值表达式，还可以是赋值表达式或函数表达式。
- 条件表达式中，表达式 1 的类型可以与表达式 2 和表达式 3 的类型不同。

If versus ?

If statement

```
#include<stdio.h>
main ()
{
   int a = 5, b = 10;
   if(a < b)
   {
      printf("b is larger!");
      printf("b is %d", b);
      b++;
      //…
   }
}
```

More space to do things!

? statement

```
#include<stdio.h>
main ()
{
   int a = 5, b = 10;

   int max = a < b ? b : a;
   printf("max is %d",max)
}
```

Can only set one variable!

If versus ?

If statement

Set multiple conditions!

```
#include<stdio.h>
main ()
{
   int a = 5, b = 10, c = 20;
   if(a < b && a < c && b < c)
   {
      printf(“c is larger!");
      printf(“c is %d“, c);
      b++;
      //…
   }
}
```

? statement

```
#include<stdio.h>
main ()
{
   int a = 5, b = 10;

   int max = a < b ? b : a;
   printf(“max is %d”,max)
}
```

Can only compare two numbers!

Nested-if

Nested if-else statement means if can be used inside another if.

```
if( condition 1 )
{
   /* code 1 */
   if( condition 2)
   {
      /* code 2 */
   }
}
```

Nested-if

Parallel if

```
     int a = 3;

     if (a > 10)
     {
1.      //...
     }else
     {
2.      //...
     }
```

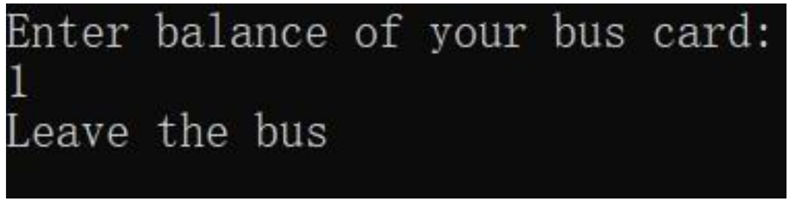
Nested if

```
        int a = 3, b = 10;

        if (a > 10)
        {
           if (b < 5)
1.   1.1   {
              //...
           }
        }else{
2.         //...
        }
```

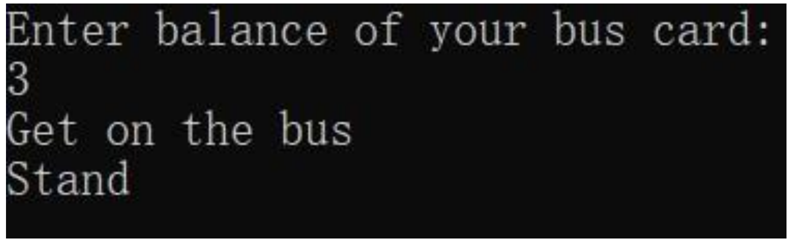
Case study: Nested-if

Case: check the balance of bus card!!!

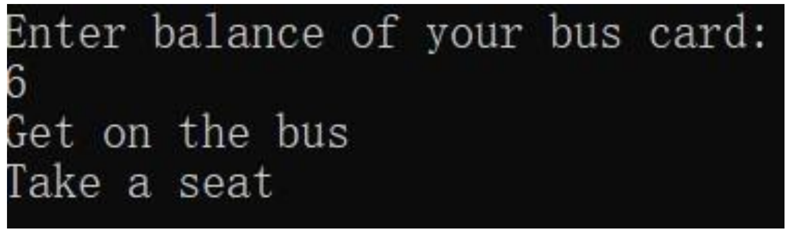
```
#include <stdio.h>
main()
{
    int a;
    printf(“Enter balance of your bus card:\n");
    scanf("%d", &a);
    if( a >= 2 )
    {
        printf(“Get on the bus\n");
        if( a >= 5 )
        {
            printf(“Take a seat\n" );
        }
        else{
            printf(“Stand");
        }
    }
    else{
        printf(“Leave the bus\n");
    }
}
```



```
Enter balance of your bus card:
1
Leave the bus
```



```
Enter balance of your bus card:
3
Get on the bus
Stand
```



```
Enter balance of your bus card:
6
Get on the bus
Take a seat
```

If statement

- 上述三种形式的if语句中在if后面都有表达式，一般为逻辑表达式或关系表达式。

```
if (expression)
        statement
```

```
if (expression)
        statement1
else
        statement2
```

```
if (expression1)
        statement1
else if (expression2)
        statement2
else
        statement3
```

- 在if和else后面可以只含有一个内嵌的操作语句，也可以由多个操作语句，此时用花括号将几个语句括起来成为一个复合语句。

✗

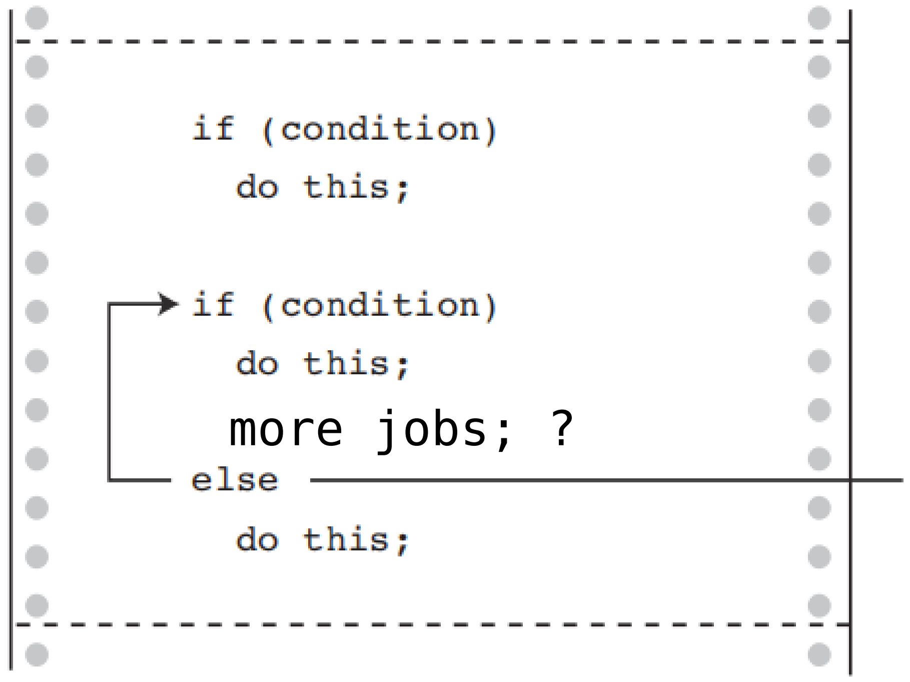
```
if (x > 0)
    printf("Incrementing x:\n");
    x++;
else              // will generate an error
    printf("x <= 0 \n");
```

√

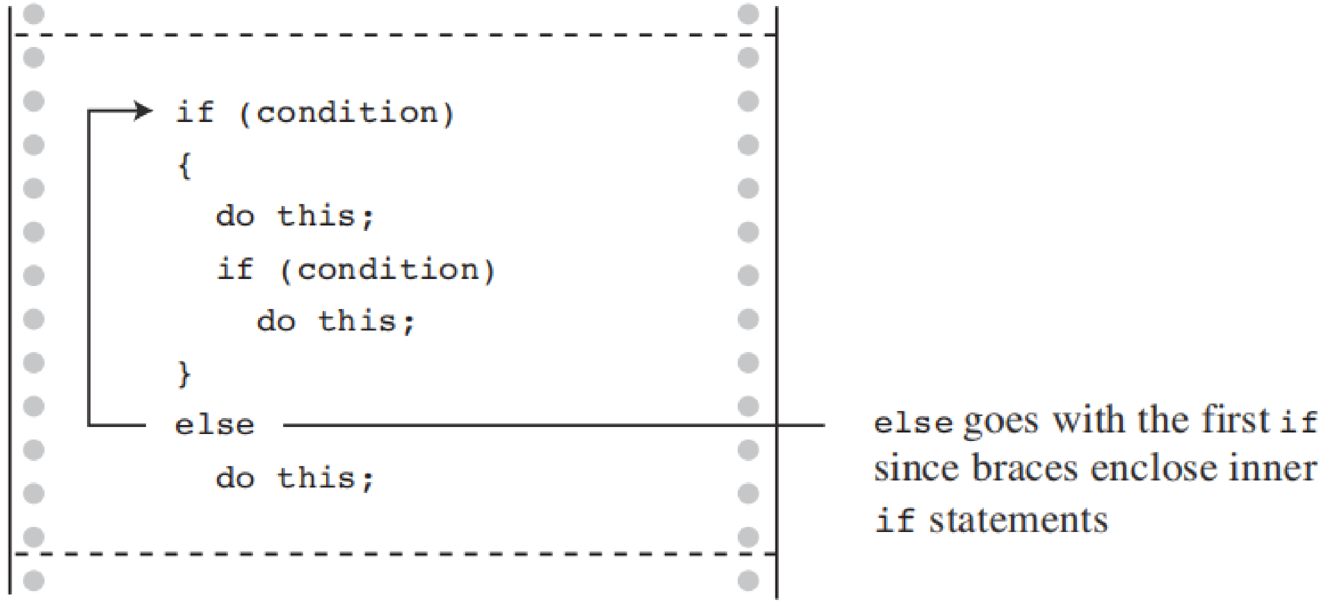
```
if (x > 0)
{
    printf("Incrementing x:\n");
    x++;
}
else
    printf("x <= 0 \n");
```

If statement

- 如果没有花括号指明，else与和它最接近的一个if相匹配



```
if (condition)
  do this;

if (condition)
  do this;
 more jobs; ?
else
  do this;
```



```
if (condition)
{
  do this;
  if (condition)
    do this;
}
else
  do this;
```

else goes with the first if since braces enclose inner if statements

If statement

- if在设置多个条件判断时需要使用逻辑运算符

```
if (90 <= range <= 100)    // NO! Don't do it!
    printf("Good show!\n");
```

✗

上述代码语义错误，会将表达式解释为：

```
(90 <= range) <= 100
```

子表达式90<=range 的值为1（真）或0（假）。因此不管range值是什么整个表达式总为真。

```
if (range >= 90 && range <= 100)
    printf("Good show!\n");
```

√

Content

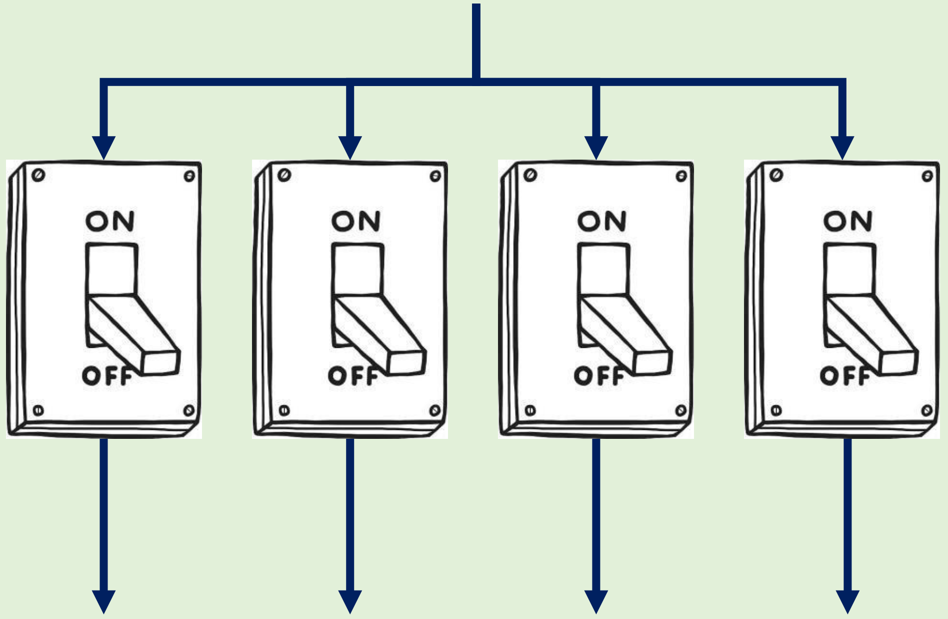
1. Logical operators
2. If statement
3. **Switch statement**

Switch statement

Switch statement allows a variable to be tested for **equality** against a list of values. Case will be switched on if equality meets

```
switch(variable)
{
   case constant1:
      statement;
      break;
   case constant2:
      statement;
      break;
   default:
      statement;
}
```

optional



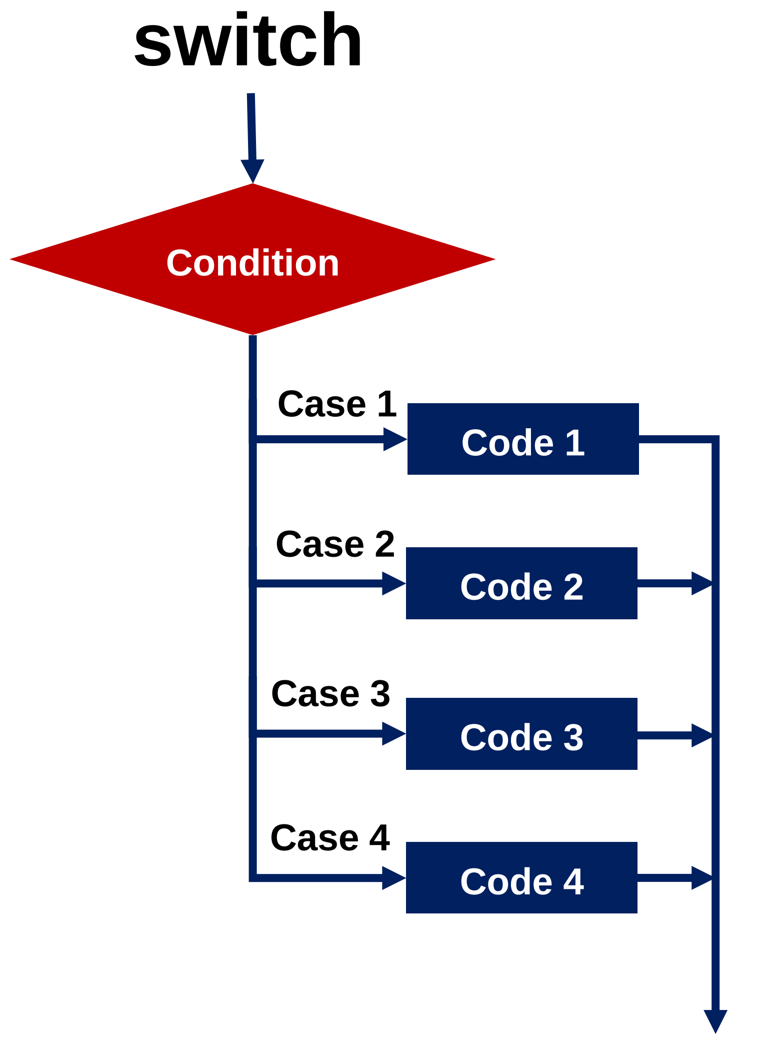
Switch versus if

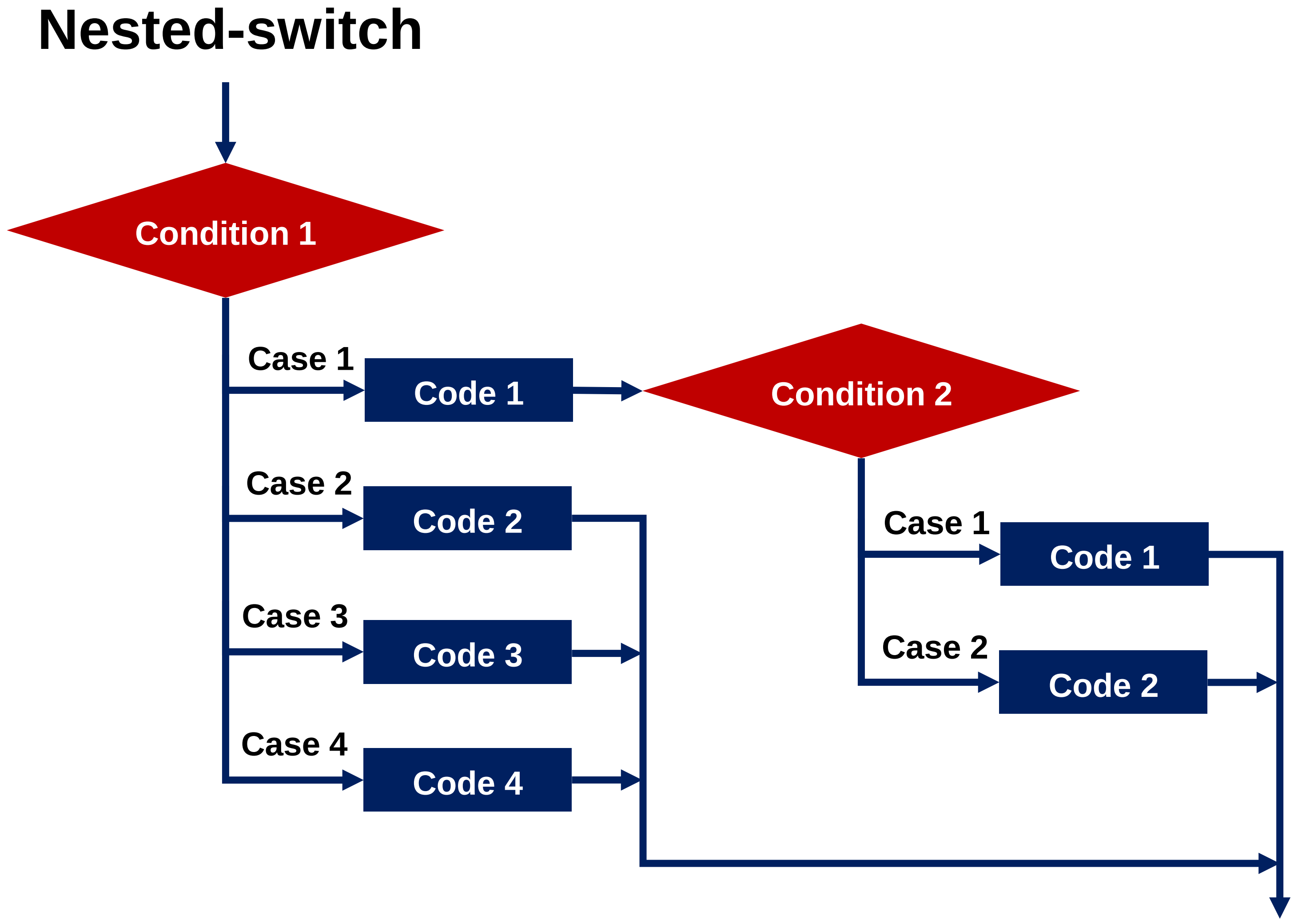
```
int a = 3;

if (a == 1)
{
   //...
}
ifelse(a == 2)
{
   //...
}else{
   //...
}
```

```
int a = 3;
switch(a)
{
  case 1:
     //...
     break;
  case 2:
     //...
     break;
  default:
     //...
}
```

Switch can only express equality!!!

Overview of switch statements

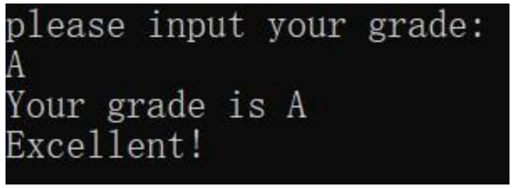




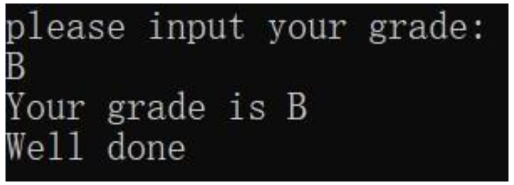
Case study: switch

Case: how to evaluate students based on grades?

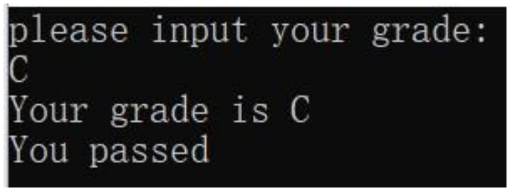
```
#include <stdio.h>
main()
{
    char a;
    printf("please input your grade:\n");
    scanf("%c", &a);
    printf("Your grade is %c\n", a );
    switch(a)
    {
        case 'A':
            printf("Excellent!\n" );break;
        case 'B':
            printf("Well done\n" );break;
        case 'C' :
            printf("You passed\n" );break;
        case 'D' :
            printf("Better try again\n" );break;
        default :
            printf("Invalid grade\n" );
    }
}
```



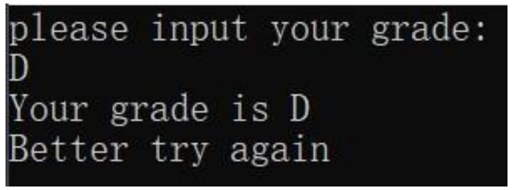
```
please input your grade:
A
Your grade is A
Excellent!
```



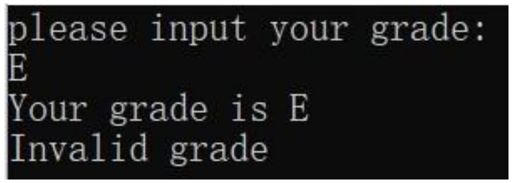
```
please input your grade:
B
Your grade is B
Well done
```



```
please input your grade:
C
Your grade is C
You passed
```



```
please input your grade:
D
Your grade is D
Better try again
```



```
please input your grade:
E
Your grade is E
Invalid grade
```

Switch statement

- switch后面括弧内的“表达式”，ANSI标准允许它为任何类型。
- 当表达式的值与某一个case后面的常量表达式的值相等时，就执行此case后面的语句，若所有的case中的常量表达式的值都没有与表达式的值匹配的，就执行default后面的语句，若都不满足则跳出。
- 每一个case的常量表达式的值必须互不相同，否则就会出现互相矛盾的现象（对表达式的同一个值，有两种或多种执行方案）。

```
int a = 3;
switch(a)
{
   case 1:
      //...
      break;
   case 2:
      //...
      break;
   default:
      //...
}
```

```
case 'A':
   printf("Excellent!\n" );break;
case ‘A':
   printf("Well done\n" );break;
```

error: duplicate case value

Switch statement

- 各个case和default的出现次序不影响执行结果。例如，可以先出现“default：…”，再出现“case ’D’：…”，然后是“case’A’：…”。

```
switch(a)
{
    case 'A':
        printf("Excellent!\n" );break;
    default :
        printf("Invalid grade\n" );
    case 'B':
        printf("Well done\n" );break;
    case 'C' :
        printf("You passed\n" );break;
    case 'D' :
        printf("Better try again\n"
    );break;
}
```

??
There is a bug. We will discuss it later.

Switch statement

- 各个case和default的出现次序不影响执行结果。例如，可以先出现“default：…”，再出现“case ’D’：…”，然后是“case’A’：…”。
- 执行完一个case后面的语句后，流程控制转移到下一 个case继续执行。“case常量表达式”只是起语句标号作用，并不是在条件判断。在执行 switch语句时，根据switch后面表达式的值找到匹配的入口标号，就从此标号开始执行下去，不再进行判断。应该在执行一个case分支后，可以用一个break语句来终止switch语句的执行。

Switch statement

```
#include<stdio.h>
int main(void){
  int grad;
  scanf("%d", &grad);
  switch(grad){
    case 4:
      printf("Excellent\n");
    case 3:
      printf("Good\n");
    case 2:
      printf("Average\n");
    case 1:
      printf("Poor\n");
    case 0:
      printf("Failing\n“);
    default:
      printf("Illegal grad\n");
  }
  return 0;}
```

```
4
Excellent
Good
Average
Poor
Failing
Illegal grad
```

```
2
Average
Poor
Failing
Illegal grad
```

Switch statement

```
#include<stdio.h>
int main(void){
  int grad;
  scanf("%d", &grad);
  switch(grad){
    case 4:
      printf("Excellent\n");
    case 3:
      printf("Good\n");
    case 2:
      printf("Average\n");
    case 1:
      printf("Poor\n");
    case 0:
      printf("Failing\n“);
    default:
      printf("Illegal grad\n");
  }
  return 0;}
```

```
4
Excellent
Good
Average
Poor
Failing
Illegal grad
```

Switch statement

```
#include<stdio.h>
int main(void){
  int grad;
  scanf("%d", &grad);
  switch(grad){
    case 4:
      printf("Excellent\n");
    case 3:
      printf("Good\n");
    case 2:
      printf("Average\n");
    case 1:
      printf("Poor\n");
    case 0:
      printf("Failing\n“);
    default:
      printf("Illegal grad\n");
  }
  return 0;}
```

```
4
Excellent
Good
Average
Poor
Failing
Illegal grad
```

Switch statement

```
#include<stdio.h>
int main(void){
  int grad;
  scanf("%d", &grad);
  switch(grad){
    case 4:
      printf("Excellent\n");
    case 3:
      printf("Good\n");
    case 2:
      printf("Average\n");
    case 1:
      printf("Poor\n");
    case 0:
      printf("Failing\n“);
    default:
      printf("Illegal grad\n");
  }
  return 0;}
```

```
4
Excellent
Good
Average
Poor
Failing
Illegal grad
```

Switch statement

```
#include<stdio.h>
int main(void){
  int grad;
  scanf("%d", &grad);
  switch(grad){
    case 4:
      printf("Excellent\n");
    case 3:
      printf("Good\n");
    case 2:
      printf("Average\n");
    case 1:
      printf("Poor\n");
    case 0:
      printf("Failing\n“);
    default:
      printf("Illegal grad\n");
  }
  return 0;}
```

```
2
Average
Poor
Failing
Illegal grad
```

Switch statement

```
#include<stdio.h>
int main(void){
  int grad;
  scanf("%d", &grad);
  switch(grad){
    case 4:
      printf("Excellent\n");break;
    case 3:
      printf("Good\n");break;
    case 2:
      printf("Average\n");break;
    case 1:
      printf("Poor\n");break;
    case 0:
      printf("Failing\n“);break;
    default:
      printf("Illegal grad\n");break;
  }
  return 0;}
```

```
4
Excellent
```

Switch statement

```
#include<stdio.h>
int main(void){
  int grad;
  scanf("%d", &grad);
  switch(grad){
    case 4:
      printf("Excellent\n");break;
    case 3:
      printf("Good\n");break;
    case 2:
      printf("Average\n");break;
    case 1:
      printf("Poor\n");break;
    case 0:
      printf("Failing\n“);break;
    default:
      printf("Illegal grad\n");break;
  }
  return 0;}
```

```
2
Average
```

Switch statement

- 各个case和default的出现次序不影响执行结果。例如，可以先出现“default：…”，再出现“case ’D’：…”，然后是“case’A’：…”。

```
switch(a)
{
    case 'A':
        printf("Excellent!\n" );break;
    default :
        printf("Invalid grade\n" );
    case 'B':
        printf("Well done\n" );break;
    case 'C' :
        printf("You passed\n" );break;
    case 'D' :
        printf("Better try again\n"
    );break;
}
```

```
??
a = ‘F’;
```

Switch statement

- 多个可以共用一组执行语句。

```
#include<stdio.h>
int main(void){
  int grad;
  scanf("%d", &grad);
  switch(grad){
    case 4:
    case 3:
    case 2:
    case 1:
      printf(“Passing\n");break;
    case 0:
      printf(“Failing\n");break;
    default:
      printf("Illegal grad\n");break;
}
  return 0;}
```

Switch statement

- 多个可以共用一组执行语句。

```
#include<stdio.h>
int main(void){
  int grad;
  scanf("%d", &grad);
  switch(grad){
    case 4: case 3: case 2: case 1:
      num_passing++;
    case 0:
      total_grades++;
      break;
}
  return 0;}
```

统计
num_passing，
total_grades

Switch statement

- 多个可以共用一组执行语句。

```
switch(ch){
    case ‘a’:
    case ‘A’: a_ct++;
              break;
    case ‘e’:
    case ‘E’: e_ct++;
              break;
    case ‘i’:
    case ‘I’: i_ct++;
              break;
    default: break;
}
```

- ☐ break出现与否结果差别较大；
- ☐ 分清程序中丢失的break是故意还是错误；

```
??constant1=‘a’ ||‘A’
```

✖ 非语法错误，‘a’ ||‘A’=0/1

```
case constant1:
    statement;
    break;
```

Switch statement

Switch statement allows a variable to be tested for **equality** against a list of values. Case will be switched on if equality meets

```
switch(variable)
{
   case constant1:
      statement;
      break;
   case constant2:
      statement;
      break;
   default:
      statement;
}
```

常量表达式constant expression，整数或者字符

5;
10+5;
'A';
✓

✗

"A"
n+1

error: case label does not reduce to an integer constant

const int n=1; ✗

#define n 1 ✓

Switch statement

Switch statement allows a variable to be tested for **equality** against a list of values. Case will be switched on if equality meets

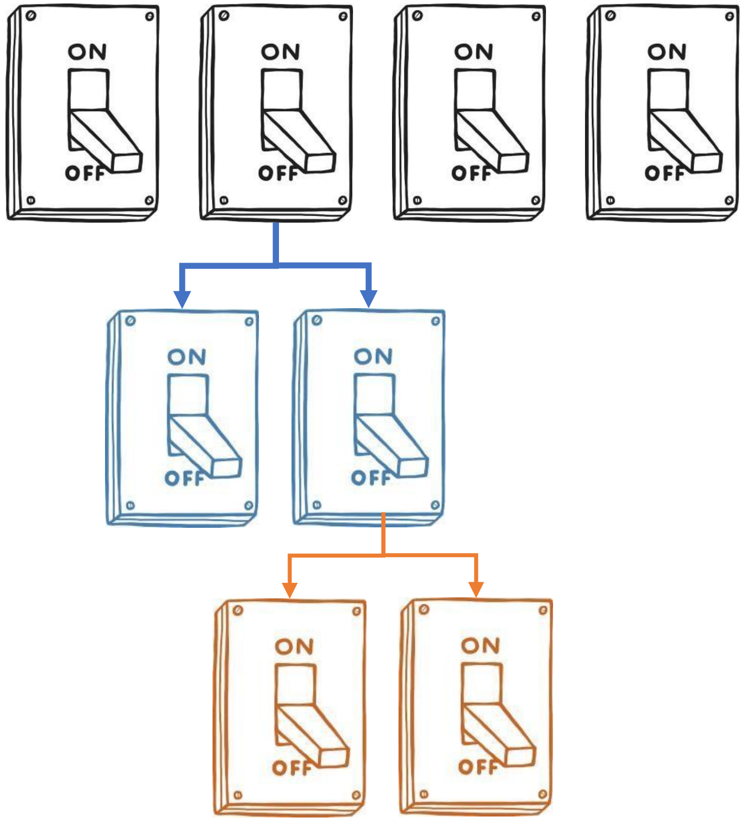
```
switch(variable)
{
   case constant2:
      {statement;
      break;}
   default:
      statement;
   case constant1:
      statement;
      break;
}
```

- ◻ { }不一定需要;
- ◻ 顺序无要求(default)

Nested-switch

Switch can be nested. Even if the case constants of the inner and outer switch are the same, no conflict will arise.

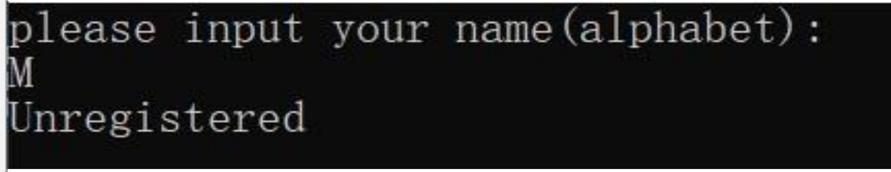
```
switch(ch1) {
case 'A’:
   switch(ch2) {
   case ‘a’:
      statement;
      break;
   case ‘A’:
      statement;
      break;
   }
case ‘B’:
}
```



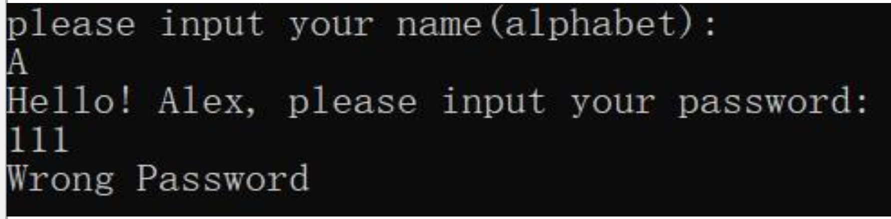
Case study: nested-switch

Case: create a simple login system!

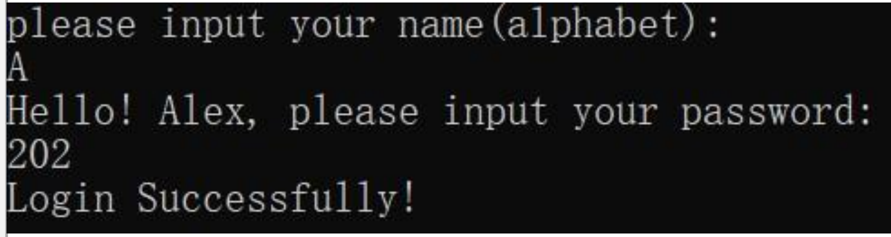
```
#include <stdio.h>
main()
{
    char a;
    int pw;
    printf("please input your name(alphabet):\n");
    scanf("%c", &a);
    switch(a) {
        case 'A':
            printf("Hello! Alex, please input your
        password:\n");
            scanf("%d", &pw);
            switch(pw) {
                case 202:
                printf("Login Successfully!");break;
                default:
                printf("Wrong Password\n");
            }break;
        default:
        printf("Unregistered\n" );
    }
}
```



```
please input your name(alphabet):
M
Unregistered
```



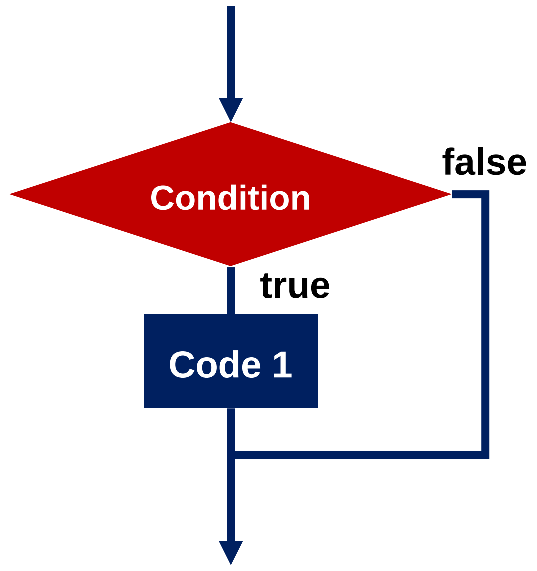
```
please input your name(alphabet):
A
Hello! Alex, please input your password:
111
Wrong Password
```



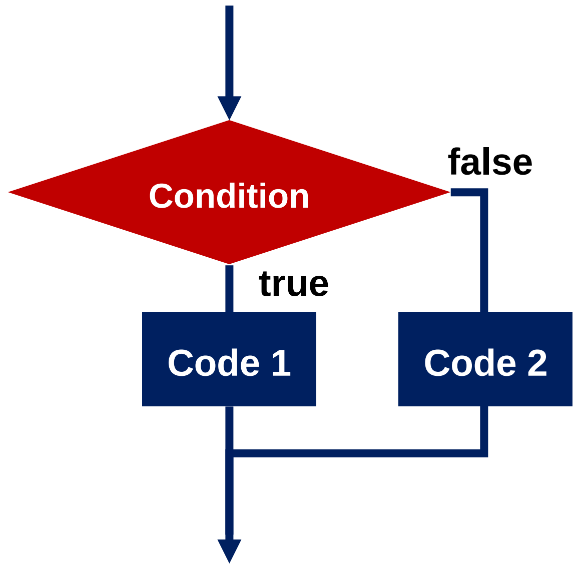
```
please input your name(alphabet):
A
Hello! Alex, please input your password:
202
Login Successfully!
```

Overview of decision-making

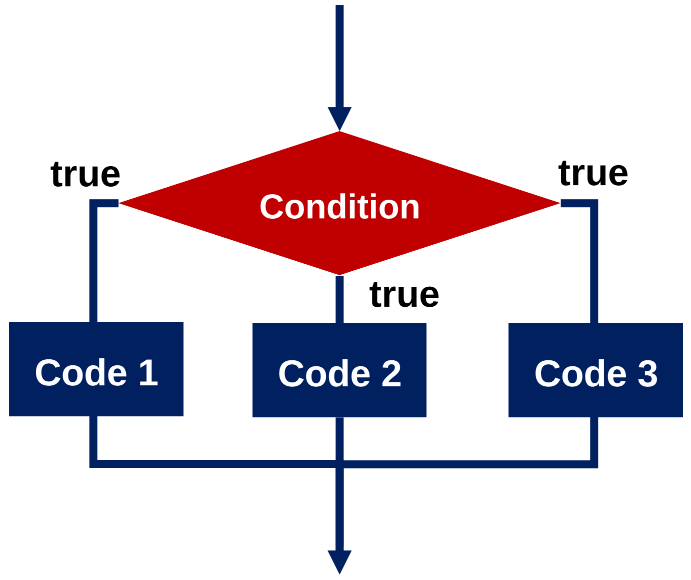
If



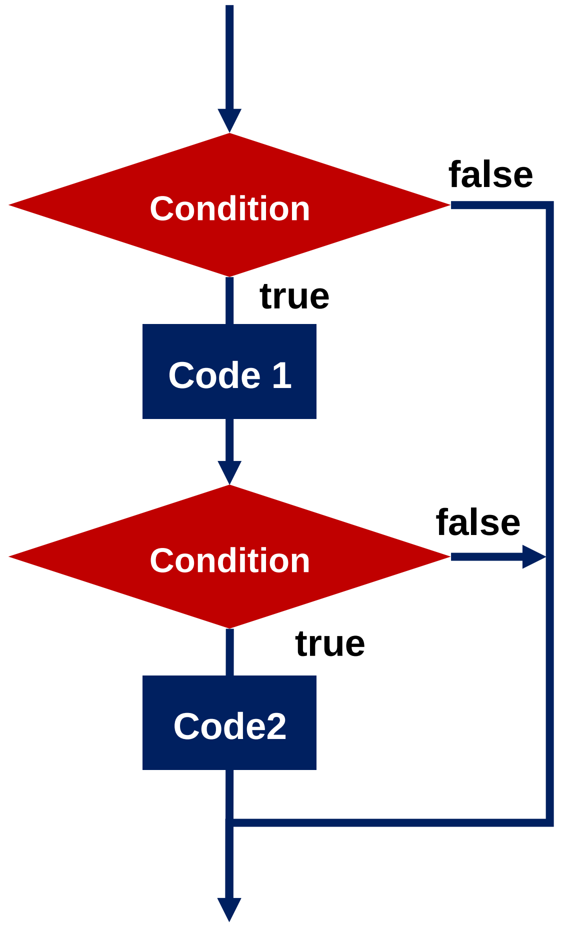
If else



If elseif



Nested if



Suppl.

Conditinal compilation

You can use them to **tell the compiler to accept or ignore blocks** of information or code according to conditions at the time of compilation.

```
(1)＃ifdef 标识符
         程序段 1
  ＃else
         程序段 2
  ＃endif
```

```
(2)＃ifndef 标识符
         程序段 1
  ＃else
         程序段 2
  ＃endif
```

```
(3) ＃if 表达式
         程序段 1
  ＃else
         程序段 2
  ＃endif
```

Conditinal compilation

- **#ifdef**指令说明，如果预处理器已定义了后面的标识符，则执行**#else**或**#endif**指令之前的所有指令并编译所有C代码（先出现哪个指令就执行到哪里）。如果预处理器未定义，且有 **#else**指令，则执行**#else**和**#endif**指令之间的所有代码。

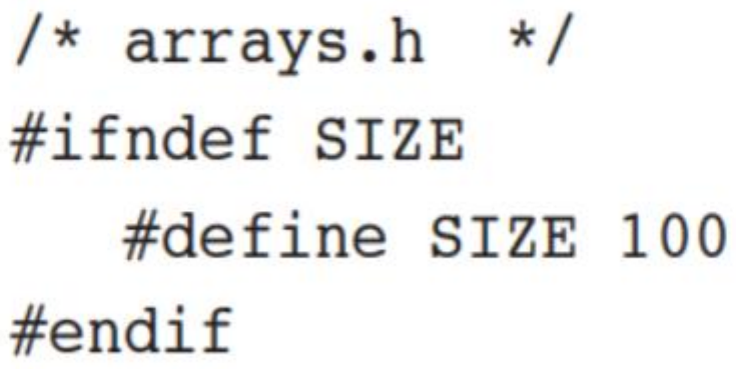
```
#ifdef 标识符
    程序段 1
#else
    程序段 2
#endif
```

```
#ifdef MAVIS
     #include "horse.h"   // gets done if MAVIS is #defined
     #define  STABLES      5
#else
     #include "cow.h"     // gets done if MAVIS isn't #defined
     #define  STABLES    15
#endif
```

Conditinal compilation

- **#ifndef**指令判断后面的标识符是否是未定义的，常用于定义之前未定义的常量：

```
#ifndef 标识符
    程序段 1
#else
    程序段 2
#endif
```



```
/* arrays.h  */
#ifndef SIZE
   #define SIZE 100
#endif
```

(Older implementations might not permit indenting the #define directive.)

- 包含多个头文件时，其中的文件可能包含了相同宏定义。**#ifndef**指令可以防止相同的宏被重复定义。

Conditinal compilation

```
#include <stdio.h>
#define LETTER 1
  void main()
     {char str[20]="C Language",c;
       int i;
       i=0;
       while((c=str[i])!='\0')
       { i++;
         #if  LETTER
            if(c>='a' && c<='z')
              c=c-32;
         #else
            if(c>='A' && c<='Z')
              c=c+32;
          #endif
          printf("%c",c);
         }
}
```

```
#if 表达式
    程序段 1
#else
    程序段 2
#endif
```

运行结果为:

C LANGUAGE

#if后面跟整型常量表达式，如果表达式为非零，则表达式为真。可以在指令中使用C的关系运算符和逻辑运算符