C程序设计基础

Introduction to C programming
Lecture 7: Array & String I

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 L6 Loop

while statement

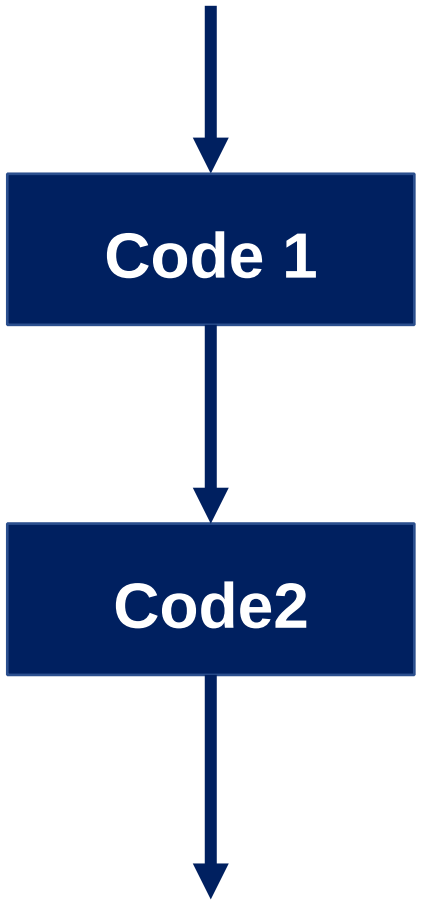
do while statement

for statement

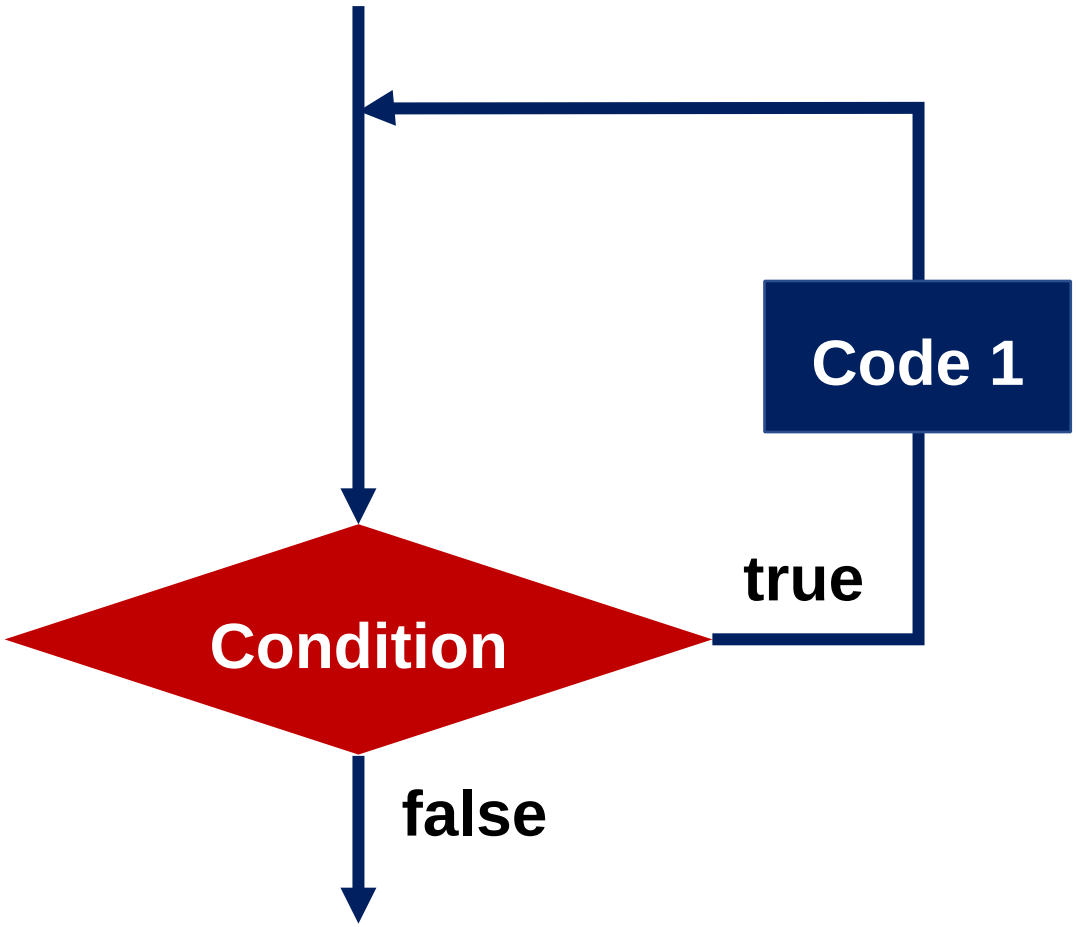
break/continue/goto

Looping in program

Sequential



Looping

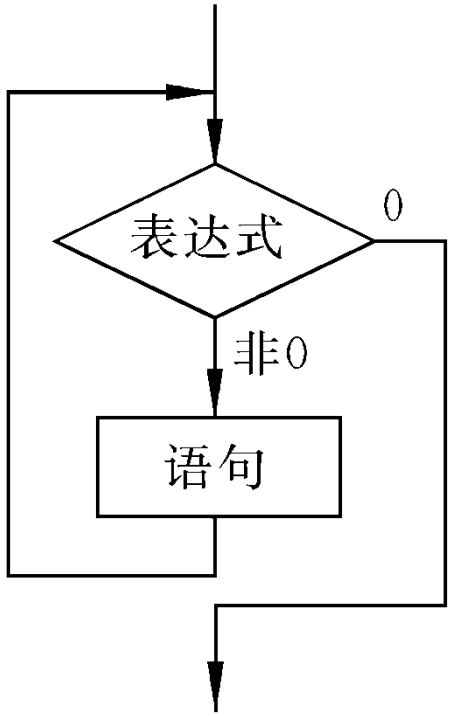


While loop

While loop repeatedly executes a statement as long as the condition is true.

```
while(condition)
    statement;
```

```
while(condition)
{
    statements;
}
```



有可能一次循环都不执行！

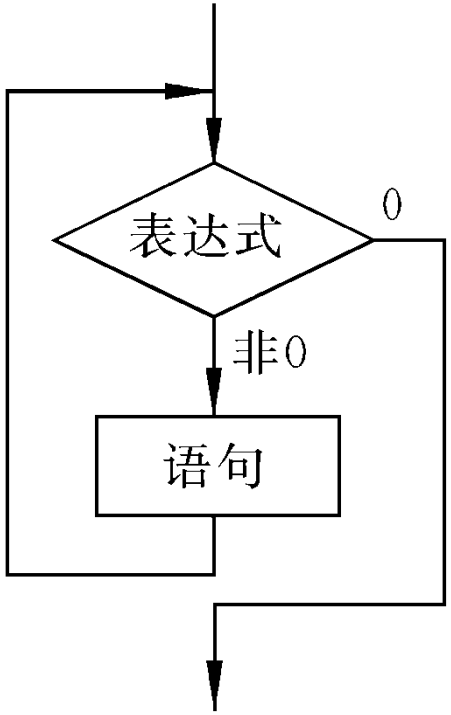
While loop

While loop repeatedly executes a statement as long as the condition is true.

```
i=1;
n=10;
while(i<n)
   i = i*2;
```

思考：i的值

```
i=1;          i is now 1
i<n成立吗？ Yes, continue
i=i*2;        i is now 2
i<n成立吗？ Yes, continue
i=i*2;        i is now 4
i<n成立吗？ Yes, continue
i=i*2;        i is now 8
i<n成立吗？ Yes, continue
i=i*2;        i is now 16
i<n成立吗？ No, exit from loop
```



While loop

```
int a = 0;
while(a < 10)
{
  // ...
  a++;
}
```

```
int a = 100;
while(a >= 10)
{
  // ...
  a--;
}
```

大小关系、
增减要匹配

While loop

Question?

What would happen?

```
while(1)
{
  // ...
}
```

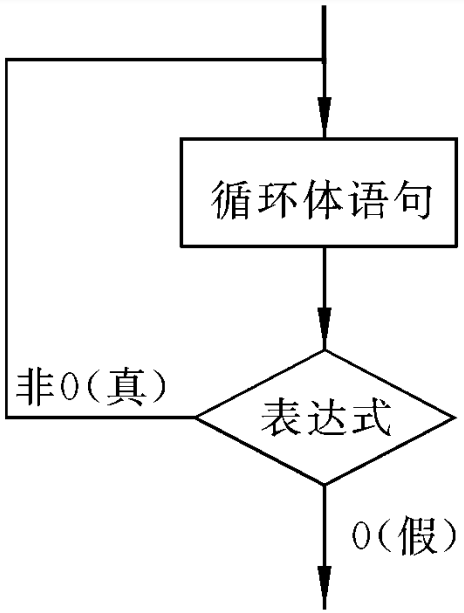
infinite loop

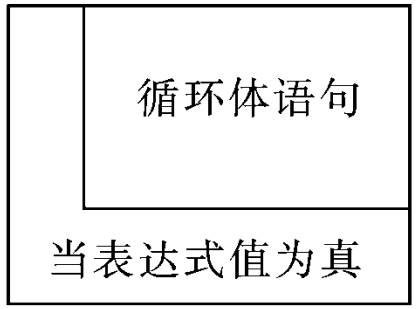
```
exit the loop with
break/goto/return/
exit
```

Do-while loop

do-while loop is similar to while loop,
it guarantees to execute **at least one time**.

```
do
{
   statements;
}while( condition );
```





(b)

至少执行一次循环！

Do-while loop

```
int a = 0;
while(a < 10)
{
  // ...
  a++;
}
```

```
int a = 0;
do
{
  // ...
  a++;
}while(a < 10);
```

while versus do-while

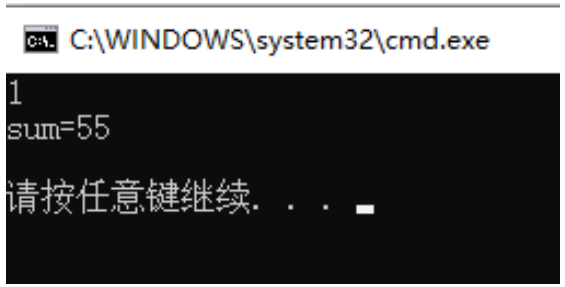
while语句和用do-while语句的比较:

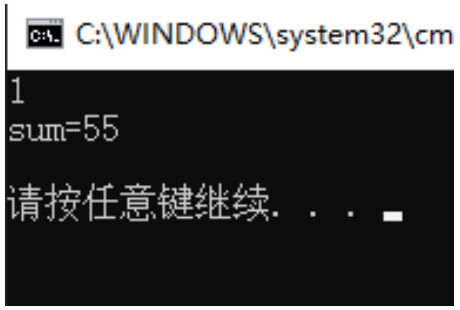
在一般情况下，用while语句和用do-while语句处理同一问题时，若二者的循环体部分是一样的，它们的结果也一样。**但是如果while后面的表达式一开始就为假(0值)时，两种循环的结果是不同的。**

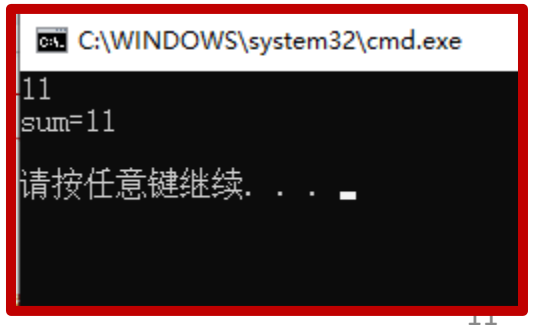
while versus do-while

```c
#include <stdio.h>
void main()
{
	int sum = 0, i;
	scanf_s("%d", &i);
	while (i <= 10)
	{
		sum = sum + i;
		i++;
	}
	printf("sum=%d\n", sum);
}
```

```c
#include <stdio.h>
void main()
{
	int sum = 0, i;
	scanf_s("%d", &i);
	do
	{
		sum = sum + i;
		i++;
	} while (i <= 10);
	printf("sum=%d\n", sum);
}
```

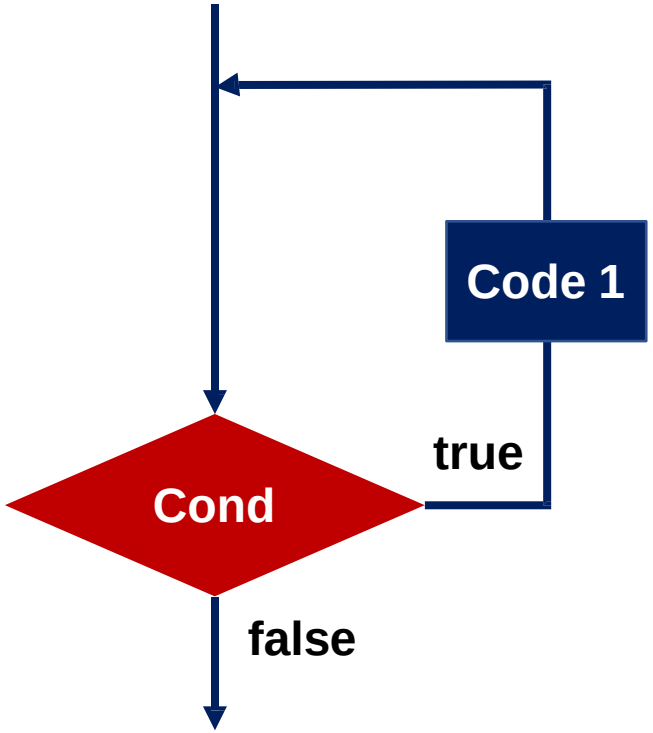




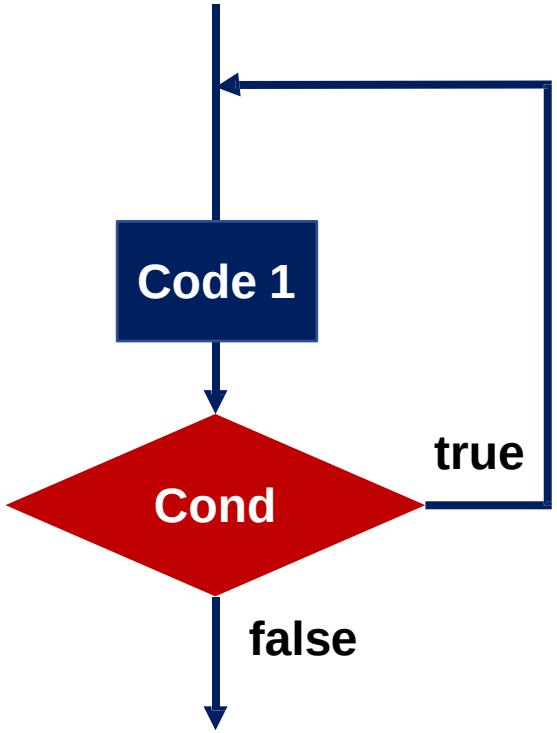


Loops

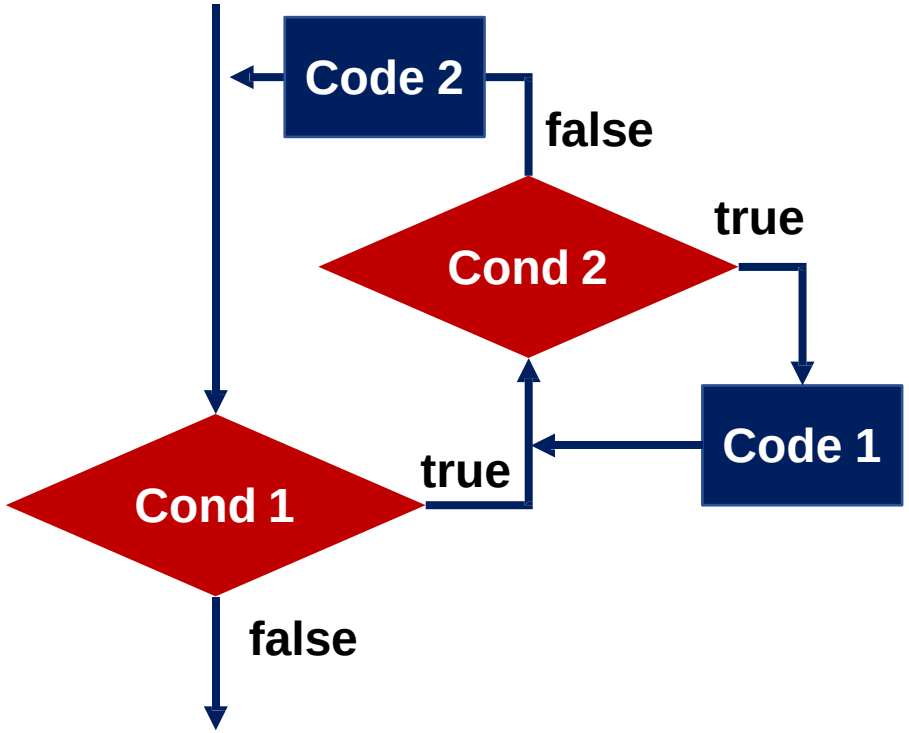
for/while loop



do-while loop



nested for loop

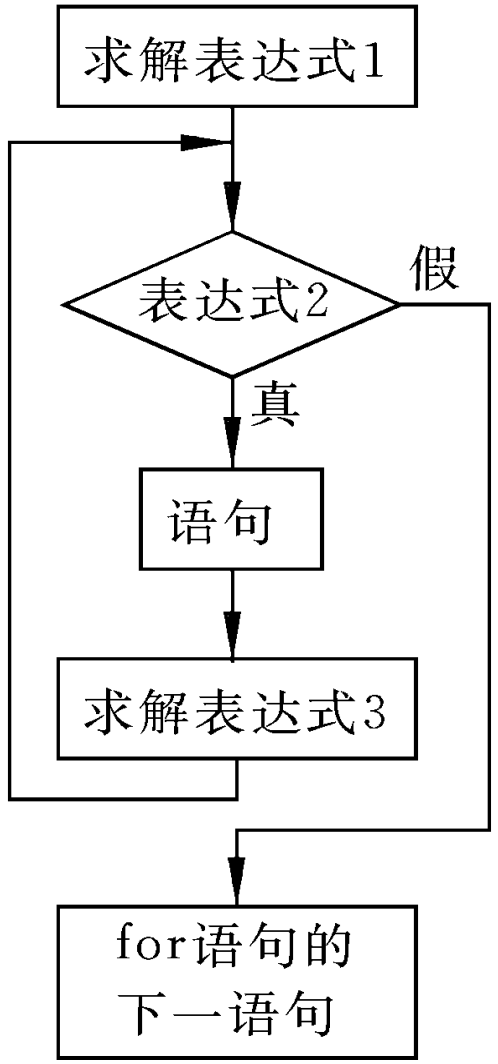


For loop

For loop is a control structure that allows repeating the same operation (but different input values) for a specific number of times.

表达式1　　表达式2　　表达式3

```
for ( init; condition; increment )
{
   statement;
}
```



For loop

计算n次

[0 n-1]

```
for(i = 0; i < n; i++)
{
  // ...
}
```

increment

[1 n]

```
for(i = 1; i <= n; i++)
{
  // ...
}
```

For loop

计算n次

[n-1 0]

```
for(i = n-1; i >=0; i--)
{
  // ...
}
```

decrement

[n 1]

```
for(i = n; i > 0; i--)
{
  // ...
}
```

For loop

易错点：

- >与<，>=与<=写反；

```
for(i = n-1; i >=0; i--)
{
  // ...
}
```

循环几次？

```
for(i = n-1; i <=0; i--)
{
  // ...
}
```

0次

For loop

易错点：

- >与<，>=与<=写反；
- <与<=，>与>=弄混；

```
for(i = n-1; i >=0; i--)
{
  // ...
}
```

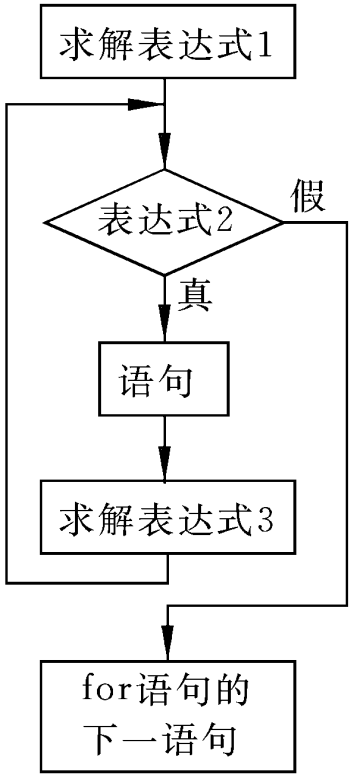
循环几次？

```
for(i = n-1; i >0; i--)
{
  // ...
}
```

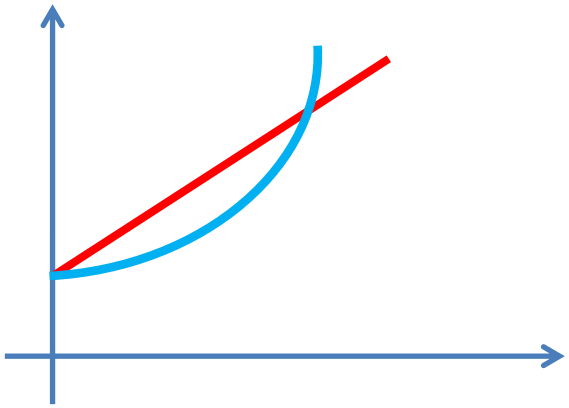
n-1次

For 作业

```c
#include<stdio.h>
int main(void)
{
  float x, y;
  int i, n;
  x = 100.f; y = 100.f;

  for(i=1; x >= y; i++){
     x +=  0.1f * 100.0f;
     y *=  (1.f + 0.05f);
     printf("i=%d, x=%f, y=%f\n", i, x, y);
    }
}
```




```
i=1, x=110.000000, y=105.000000
i=2, x=120.000000, y=110.250000
i=3, x=130.000000, y=115.762497
...
i=25, x=350.000000, y=338.635406
i=26, x=360.000000, y=355.567169
i=27, x=370.000000, y=373.345520
```



Nested loops

C allows using one loop inside another loop.

```
while ()
{
   // xxxx
   while()
   {
      // xxxx
   }
}
```

```
do
{
   // xxxx
   do
   {
      // xxxx
   }while();
}while();
```

```
for ( ; ;  )
{
   for ( ; ;  )
   {
      // xxxx
   }
}
```

Nested loops

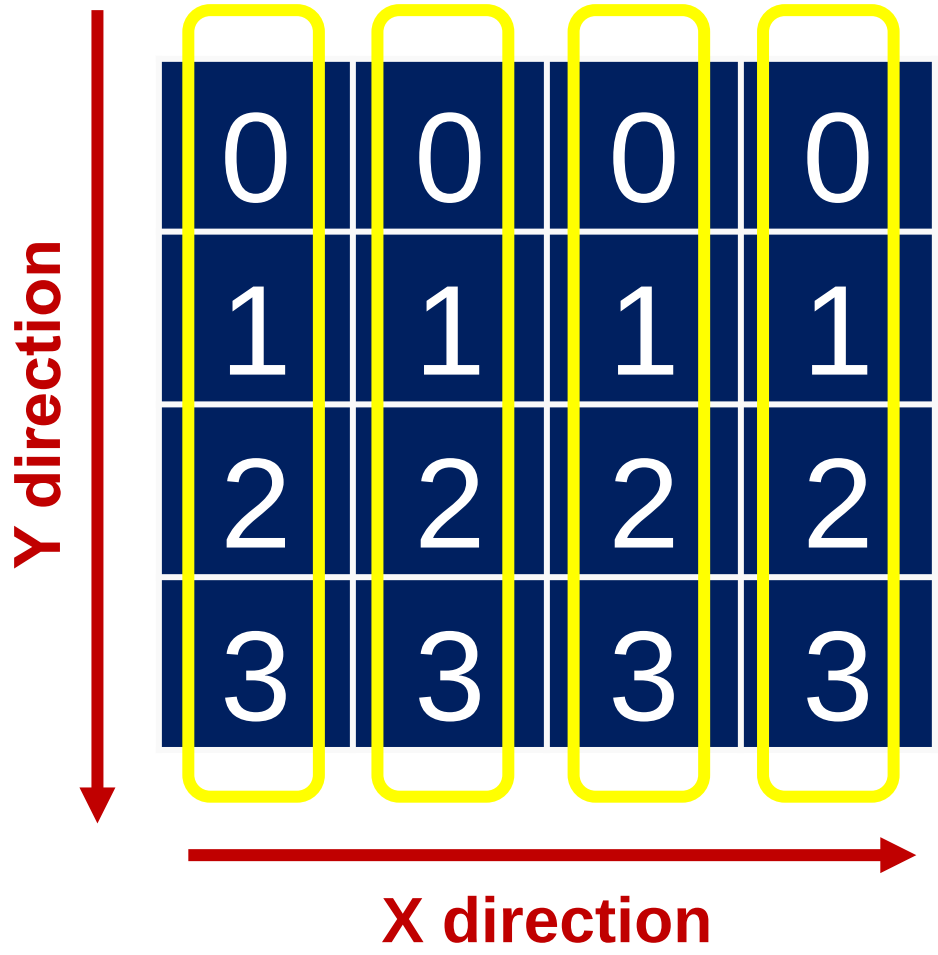
C allows using one loop inside another loop.

```
while ()
{
   // xxxx
   do
   {
      // xxxx
   }while()
}
```

```
do
{
   // xxxx
   for (; ;)
   {
      // xxxx
   }
}while();
```

```
for ( ; ;  )
{
   while()
   {
      // xxxx
   }
}
```

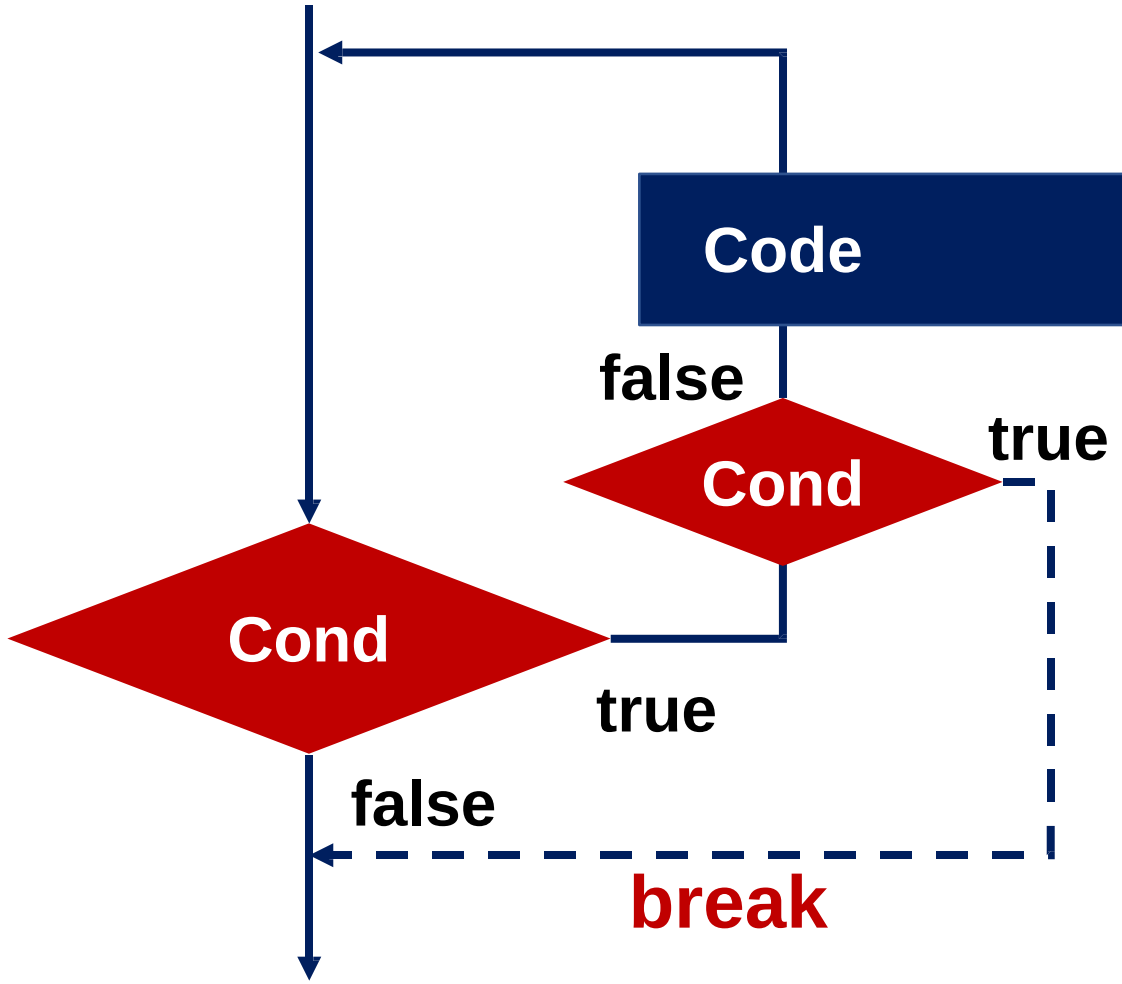
Nested loops



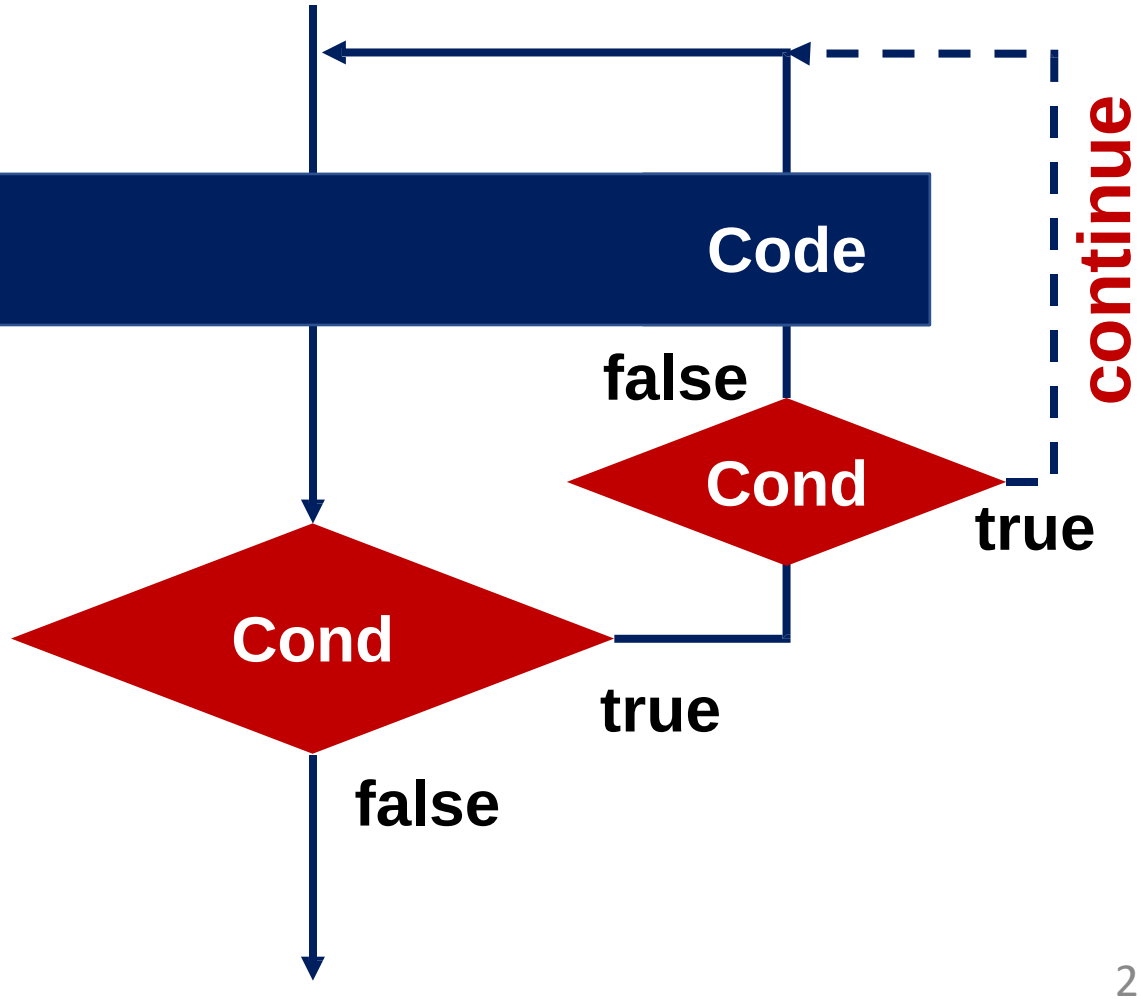

```
for (int x = 0; x < 4; x++)
{
    for (int y = 0; y < 4; y++)
    {
        // fill y at <x, y>
    }
}
```

Recap last lectures

Break loop



Continue loop



goto

goto跳到**同一**函数中任何有标号（identifier）的地方。

```
…
identifier : statement
…
goto identifier;
```

goto

```
while (…){
  switch (…){
    …
    goto loop_done
    …
  }
}
loop_done: …
```

其他方案？

不是所有的goto 都可以用break代替

- 少用goto（容易混乱）
- 与continue, break, exit, return等混合使用

Objective of this lecture

You can use array to process a group of data!

Content

1. 1-D array
2. 2-D and N-D array
3. String
4. Row-major or column-major order

Content

1. 1-D array
2. 2-D and N-D array
3. String
4. Row-major or column-major order

1-D array in life

1-D array in life

1-D array in life

1-D array in life

Why do we need array?

One student

```
char name = 'J';
int age = 18;
float height = 1.75;
```

Many students

```
char name1 = 'J'; int age1 = 18; float height1 = 1.75;
char name2 = 'R'; int age2 = 19; float height2 = 1.82;
char name3 = 'M'; int age3 = 18; float height3 = 1.72;
char name4 = 'T'; int age4 = 20; float height4 = 1.85;
…
```

You need to declare many variables!!!

Why do we need array?

```
main()
{
    float student_1;
    float student_2;
    float student_3;
    …
    float student_30;

    scanf("%f", &student_1);
    scanf("%f", &student_2);
    scanf("%f", &student_3);
    …
    scanf("%f", &student_30);
}
```

For loop cannot solve the problem!!!

```
main()
{
    for (int i = 0; i < 30; i++)
    {
        float student_i;
        scanf("%f", &student_i);
    }
}
```

Why do we need array?

Is there any way of solving these problems?

Let’s go on our journey to the array world!

1-D array

C provides a data structure called **array**. It stores a fixed-size collection of elements of the same type.

```
type name[size] = {…};

type name[] = {…};
```

int array

3	2	1	5	…	8

float array

1.2	4.5	-1.9	3.4	…	8.8

char array

H	R	O	Y	…	P

1-D array

Declare, initialize and access an int array:

- int a[10]; **// declare**
- a[0] = 3, a[1] = 2, …., a[9] = 7; **// initialize**
- int a[10] = {3, 2, 1, 5, 6, 8, 9, 2, 0, 7}; **// declare and initialize**
- int a[] = {3, 2, 1, 5, 6, 8, 9, 2, 0, 7}; **// declare and initialize**
- printf("a[5] = %d", a[5]); **// access the array**

Declare

数组名定名规则和变量名相同，遵循标识符定名规则：

1）变量名的开头必须是字母或下划线，不能是数字。实际编程中最常用的是以字母开头，而以下划线开头的变量名是系统专用的。

所以为了避免与系统定义的名字产生冲突，在编程的时候，除非要求这么定义，否则永远都不要使用下划线作为一个变量名的开头。

2）变量名中的字母是区分大小写的。比如 a 和 A 是不同的变量名，num 和 Num 也是不同的变量名。

3）变量名绝对不可以是C语言关键字，这一点一定要记住！

4）变量名中不能有空格。这个可以这样理解：因为上面我们说过，变量名是字母、数字、下划线的组合，没有空格这一项。

Declare

在定义数组时，需要指定数组中元素的个数，方括弧中的常量表达式用来表示元素的个数，即数组长度。

- √ int a[10];
- √ int a[] = {3, 2, 1, 5, 6, 8, 9, 2, 0, 7};
- √ int a[10] = {1};
- × int a[];

Declare

常量表达式中可以包括常量和符号常量，但不能包含变量。C语言不允许对数组的大小作动态定义，即数组的大小不依赖于程序运行过程中变量的值。

```
int n;
scanf("%d", &n);    /*在程序中临时输入数组的大小*/
```

但是可使用宏定义来实现直接替换：

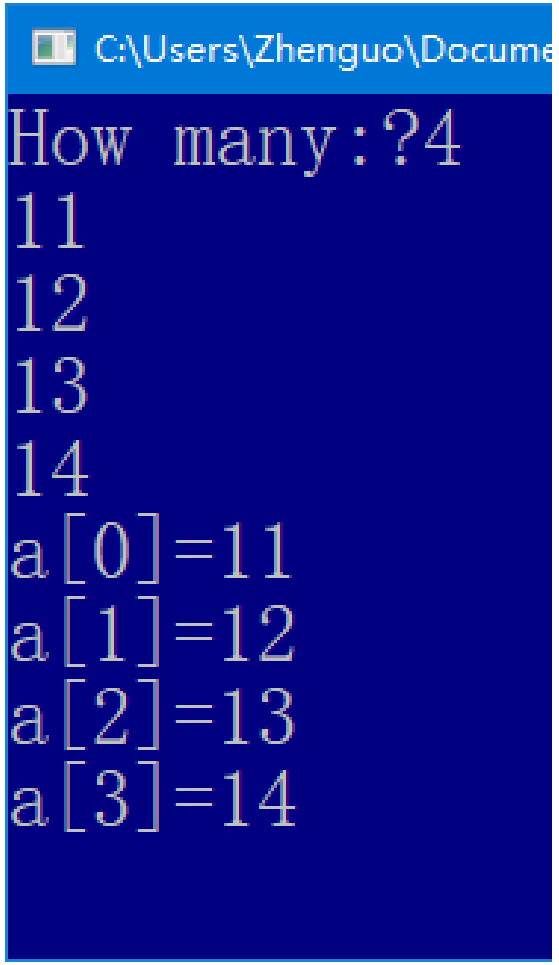
However…

```
#define N  10;
int a[N];
```

Declare

```
#include <stdio.h>
main() {
     int i, n;
     printf("How many:?");
     scanf("%d", &n);

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

     for (i = 0; i < n; i++) {
          printf("a[%d]=%d\n", i, a[i]);
     }
     return 0;
}
```

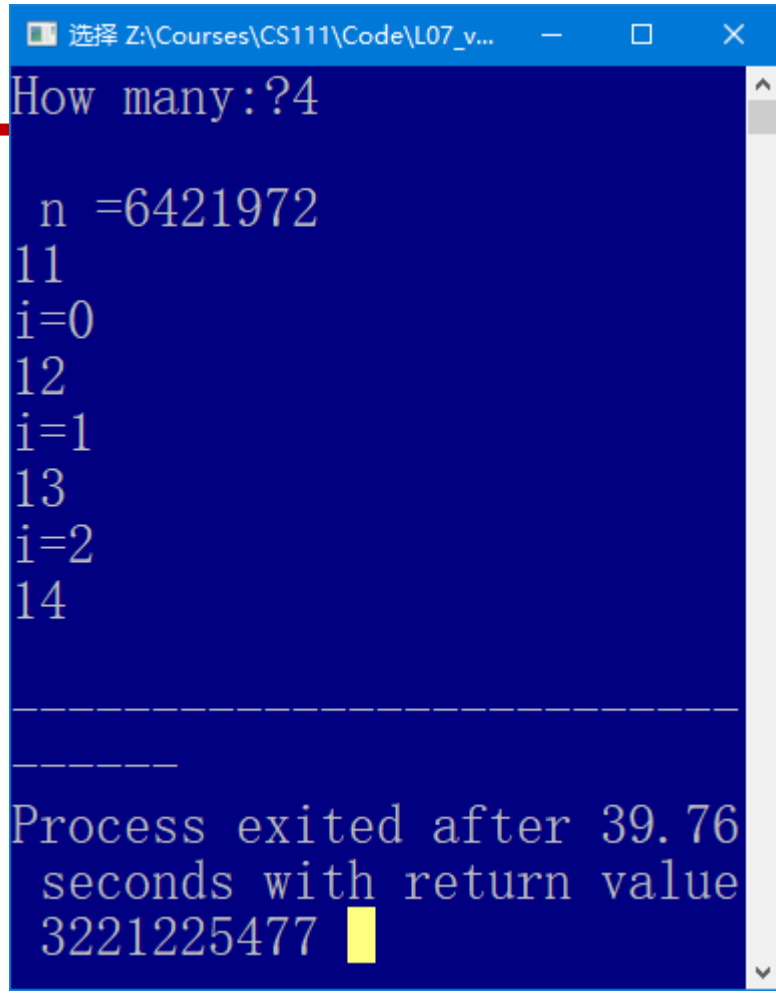
C99可实现变长数组，在函数调用中有诸多好处…



Declare

```
#include <stdio.h>
main() {
     int i, n;
     printf("How many:?");
     int a[n];
     scanf("%d", &n);
     printf("\n n=%d\n", &n);
     for (i = 0; i < n; i++){
          scanf("%d", &a[i]);
          printf("i=%d\n", i);
     }
     for (i = 0; i < n; i++) {
          printf("a[%d]=%d\n", i, a[i]);
     }
     return 0;}
```

C99可实现变长数组，在函数调用中有诸多好处，但是定义须在给定n之后



Declare

其他常见的错误：

① `float a[0];` /* 数组大小为0没有意义 */

② `int b(2)(3);` /* 不能使用圆括号 */

③ `int k, a[k];` /* 不能用变量说明数组大小*/

Initialize

1. 在定义数组时对数组元素赋以初值。

int a[10] = {3, 2, 1, 5, 6, 8, 9, 2, 0, 7}; // length is 10

{ }: array initializer(数组初始化器)

3	2	1	5	6	8	9	2	0	7
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]

Index starts at 0，[ ]取下标/索引

Can we access array by a[10]?

Initialize

1. 在定义数组时对数组元素赋以初值。

```
int a[10] = {3, 2, 1, 5, 6, 8, 9, 2, 0, 7}; // length is 10
```

3	2	1	5	6	8	9	2	0	7
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]

C数组语言下标从0开始，其他语言如Python亦如此，但是Fortran和Matlab则是从1开始。

Initialize

2. 可以只给一部分元素赋值。

```
int a[10] = {3, 2, 1}; // length is 10, fit rests with 0
```

3	2	1	0	0	0	0	0	0	0
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]

Initialize

3. 在对全部数组元素赋初值时，由于数据的个数已经确定，因此可以不指定数组长度。

```
int a[] = {3, 2, 1}; // length is 3
```

3	2	1
a[0]	a[1]	a[2]

Initialize

int a[10] = {3, 2, 1, 5, 6, 8, 9, 2, 0, 7, **4**}; // length is 10

Fixed size

Wrong! Exceed the length

int a[] = {3, 2, 1, 5, 6, 8, 9, 2, 0, 7, **4**}; // length is 11

Size not fixed at declaration

Correct

Initialize

4. 如果想使一个数组中全部元素值为0，可以写成:

```
int  a [10] ={0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
```

或者

```
int  a [10] ={0};
```

有时 int a[10];
数组a中的默认值一般为零，可能取决于编译器，
但为了确保其结尾零，可用以上的操作实现。

Initialize

5. 指示器

```
int  a [15] ={0, 0, 29, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 48};

int  a [15] ={[2]=29,  [9]=7,  [14]=48};

int  a [15] ={[14]=48 , [9]=7,  [2]=29};

int  a [15] ={0, 0,  29,  [14]=48 , [9]=7};
```

Initialize

5. 指示器

```
int  a [ ] ={[5]=10, [23]=13, [11]=36, [15]=29 };
```

len = 24

1-D array

You can also define float array and char array

float array: float a[] = {1.2, -0.6, 1000, -32, 5.34};

1.2	-0.6	1000	-32	5.34

char array: char c[] = {‘h’, ‘e’, ‘l’, ‘l’, ‘o’, ‘!’};

‘h’	‘e’	‘l’	‘l’	‘o’	‘!’

1-D array

char array: char c[5] = {‘h’, ‘e’, **2, 2.3**, ‘o’}; **// Wrong! Must be in same type!**

int array: int c[5] = {0, 1, 2, **2.5**, 5}; **// Wrong! Must be in same type!**

```
#include<stdio.h>
main()
{
    char c[5] = { 'h','e',2,2.3,'o' };
    printf("%f",c[3]);
}
```


```
#include<stdio.h>
main()
{
    int c[5] = {0,1,2,2.5,5};
    printf("%f",c[3]);
}
```

注意：只能引用数组元素而不能一次整体调用整个数组全部元素

1-D array

Question:

float a[5] = {1.0, 2.0, 3, 4.0, 5.};

Is this legal? And why?

What are outputs of the following codes?

```
int k;
float b[5] = {1., 2., 3, 4., 5.};
for(k = 0; k < 5; k ++)
    printf("%f\n", b[k]);
```

1-D array

char array: char c[5] = {‘h’, ‘e’, **2, 2.3**, ‘o’}; **// Wrong! Must be in same type!**

int array: int c[5] = {0, 1, 2, **2.5**, 5}; **// Wrong! Must be in same type!**

注意：
与机器、编译器有关有关，
尽量写成标准形式

1-D array

与循环结合的易错点：

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

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

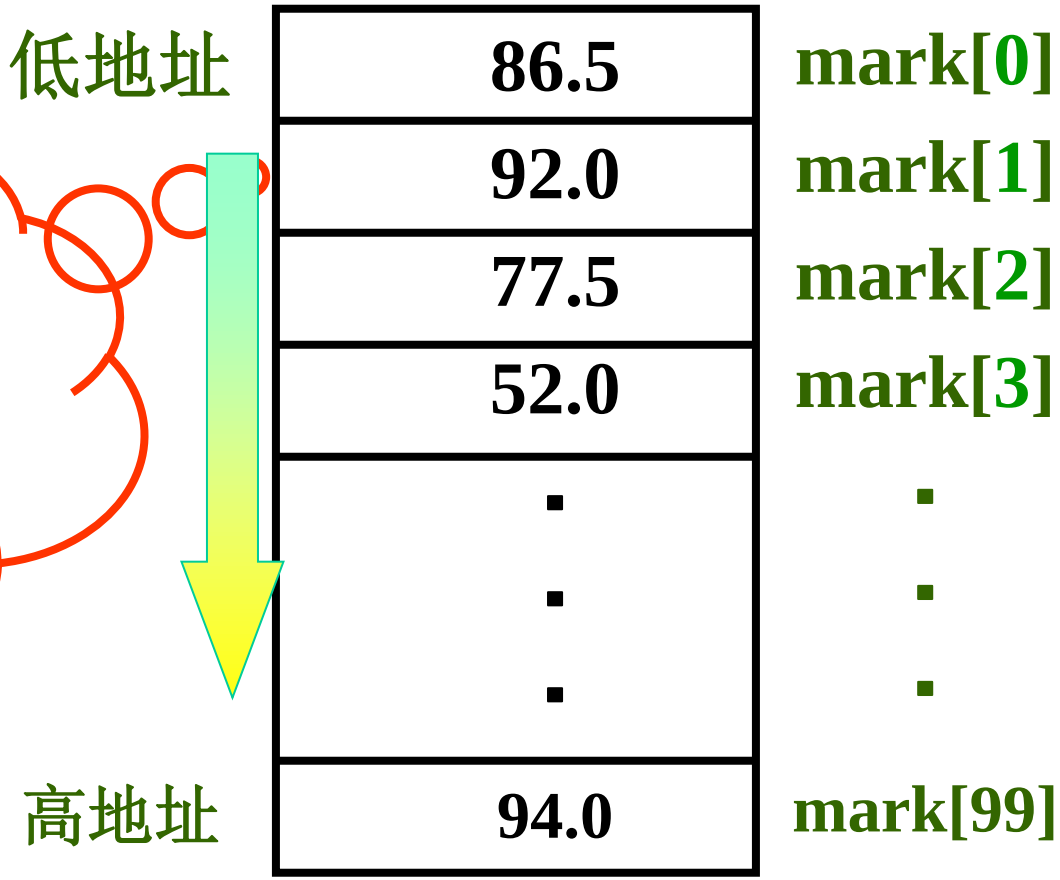
```
for(i=0; i<N; i++)
  a[i]=b[i];
```

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

1-D array

每个数据元素占用的字节数，就是基类型的字节数

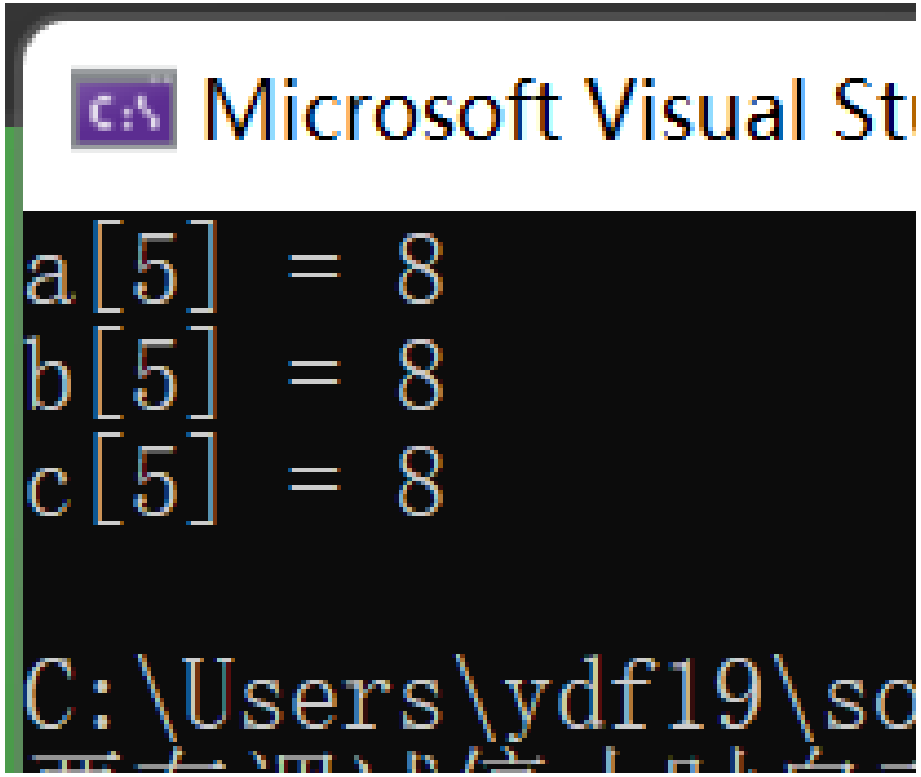
一个元素占4个字节



Case study: 1-D array

```
main()
{
int a[10];
a[0] = 3;
a[1] = 2;

…………

a[9] = 7;

int b[10] = { 3,2,1,5,6,8,9,2,0,7 };

int c[] = { 3,2,1,5,6,8,9,2,0,7 };

printf("a[5] = %d\n", a[5]);
printf("b[5] = %d\n", b[5]);
printf("c[5] = %d\n", c[5]);
}
```

Case: declare, initialize and access an int array



Case study: 1-D array

Case: if you want to measure temperature of 10 persons?

```
main()
{
    float temperature[10];

    for (int i = 0; i < 10; i++)
        scanf("%f", &temperature[i]);

    for (int i = 0; i < 10; i++)
        printf("%f ", temperature[i]);
}
```

Microsoft Visual Studio 调试控制台

```
36.5 36.5 36.5 36.5 36.5 36.5 36.5 36.5 36.5 36.5
36.500000 36.500000 36.500000 36.500000 36.500000 36.500000
 36.500000 36.500000 36.500000 36.500000
C:\Users\ydf19\source\repos\Hello\x64\Debug\Hello.exe (进程
```

Case study: 1-D array

Case: how many people are on the train：Calculate the total number of people

```
main()
{
    int carriages[6] = {34,56,89,32,76,39};
    int all = 0;
    for (int i = 0; i < 6; i++)
          all = all + carriages[i];
    printf("There are %d people on the train",
    all);
}
```

You can also use “all += carriages[i];”

Microsoft Visual Studio 调试控制台

```
There are 326 people on the train
```

Case study: 1-D array

Case: scanf and printf a string.

```
main()
{
    char c[8] = {'S','U','S','T','e','c','h'};

    printf("%s\n",c);

    for (int i = 0; i < 8; i++)
        printf("%c",c[i]);
}
```

Operations of 1-D array

int a[10] = {3, 2, 1, 5, 6, 8, 9, 2, 0, 7};

3	2	1	5	6	8	9	2	0	7

```
b = a
```

```
int b[] = a;
```

数组变量本身不能被赋值

```
for(i=0;i<n;i++)
{
  b[i] = a[i];
}
```

Operations of 1-D array

int a[10] = {3, 2, 1, 5, 6, 8, 9, 2, 0, 7};

3	2	1	5	6	8	9	2	0	7

+

int b[10] = {2, 7, 2, 3, 4, 1, 1, 1, 3, 5};

2	7	2	3	4	1	1	1	3	5

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
+	+	+	+	+	+	+	+	+	+
b[0]	b[1]	b[2]	b[3]	b[4]	b[5]	b[6]	b[7]	b[8]	b[9]
5	9	3	8	10	9	10	3	3	12

Operations of 1-D array

int a[10] = {3, 2, 1, 5, 6, 8, 9, 2, 0, 7};

3	2	1	5	6	8	9	2	0	7

- int b[10] = {2, 7, 2, 3, 4, 1, 1, 1, 3, 5};

2	7	2	3	4	1	1	1	3	5

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
-	-	-	-	-	-	-	-	-	-
b[0]	b[1]	b[2]	b[3]	b[4]	b[5]	b[6]	b[7]	b[8]	b[9]
1	-5	-1	2	2	7	8	1	-3	2

Operations of 1-D array

int a[10] = {3, 2, 1, 5, 6, 8, 9, 2, 0, 7};

3	2	1	5	6	8	9	2	0	7

*

int b[10] = {2, 7, 2, 3, 4, 1, 1, 1, 3, 5};

2	7	2	3	4	1	1	1	3	5

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
*	*	*	*	*	*	*	*	*	*
b[0]	b[1]	b[2]	b[3]	b[4]	b[5]	b[6]	b[7]	b[8]	b[9]
6	14	2	15	24	8	9	2	0	35

Operations of 1-D array

float a[10] = {3, 2, 1, 5, 6, 8, 9, 2, 0, 7};

3	2	1	5	6	8	9	2	0	7

/

float b[10] = {2, 7, 2, 3, 4, 1, 1, 1, 3, 5};

2	7	2	3	4	1	1	1	3	5

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
/	/	/	/	/	/	/	/	/	/
b[0]	b[1]	b[2]	b[3]	b[4]	b[5]	b[6]	b[7]	b[8]	b[9]
1.5	0.28	0.5	1.67	1.5	8	9	2	0	1.4

Case study: calculations

Case: make four basic operations between two integer arrays.

```
main()
{
   int a[10] = {3,2,1,5,6,8,9,2,0,7};
   int b[10] = {2,7,2,3,4,1,1,1,3,5};
   int c[10], d[10], e[10];
   float f[10];

   for (int i = 0; i < 10; i++)
   {
      c[i] = a[i] + b[i];
      d[i] = a[i] - b[i];
      e[i] = a[i] * b[i];
      f[i] = (float)a[i] / b[i];
   }

   for (int i = 0; i < 10; i++)
   {
      printf("%d  %d  %d  %f\n", c[i],d[i],e[i],f[i]);
   }
}
```

Microsoft Visual Studio 调试

```
5   1   6   1.500000
9   -5   14   0.285714
3   -1   2   0.500000
8   2   15   1.666667
10   2   24   1.500000
9   7   8   8.000000
10   8   9   9.000000
3   1   2   2.000000
3   -3   0   0.000000
12   2   35   1.400000
```

Operations of 1-D array: sorting

2020年中国大学排行榜

www.cnur.com

排名	高校名称	省市	总分
1	北京大学	北京	100.0
2	清华大学	北京	99.9
3	中国科学技术大学	安徽	97.2
4	南京大学	江苏	96.5
5	复旦大学	上海	94.7
6	中国人民大学	北京	94.3
7	浙江大学	浙江	93.8
8	上海交通大学	上海	93.1
9	哈尔滨工业大学	黑龙江	91.9
10	西安交通大学	陕西	91.7
11	南开大学	天津	91.6
12	武汉大学	湖北	91.0
13	中山大学	广东	90.8
14	东南大学	江苏	90.3
15	厦门大学	福建	89.7
16	同济大学	上海	89.6
17	华中科技大学	湖北	89.4
18	北京航空航天大学	北京	87.8
19	天津大学	天津	87.6
20	北京理工大学	北京	87.1
21	北京师范大学	北京	86.9
22	国防科技大学	湖南	86.5
23	中国科学院大学	北京	86.4
24	大连理工大学	辽宁	85.3
25	西北工业大学	陕西	85.1
26	吉林大学	吉林	84.9
27	四川大学	四川	84.2
28	兰州大学	甘肃	83.6
29	山东大学	山东	83.5
30	电子科技大学	四川	82.0

中国大学排行榜 www.cnur.com

We love ranking!!!

61	Saudi Arabia	1386	1364	22	▲
68	United Arab Emirates	1362	1330	32	▲
70	Iraq	1355	1353	2	▼
71	China PR	1353	1323	30	▲
79	Oman	1305	1302	3	▲
80	Syria	1303	1304	-1	▼

Operations of 1-D array: sorting

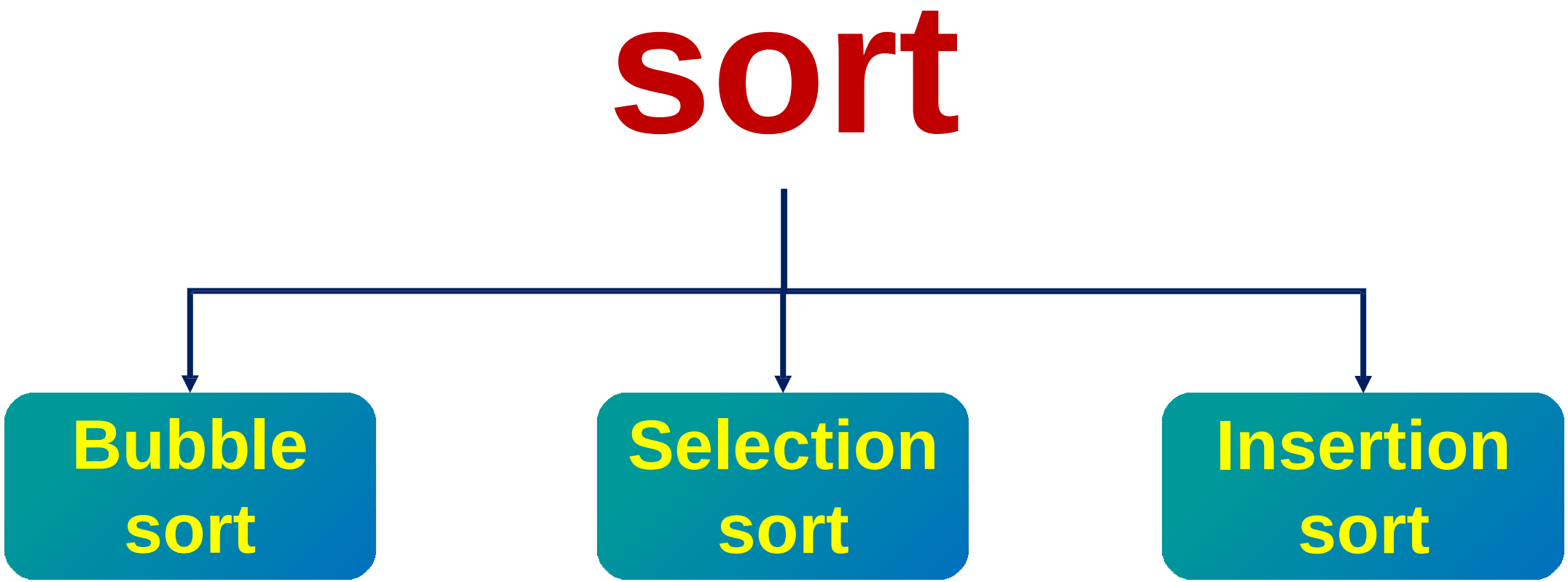
int a[10] = {3, 2, 1, 5, 6, 8, 9, 2, 0, 7};

3	44	38	5	47	15	36	26	27	2	46	4	19	50	48

How the sort the array?

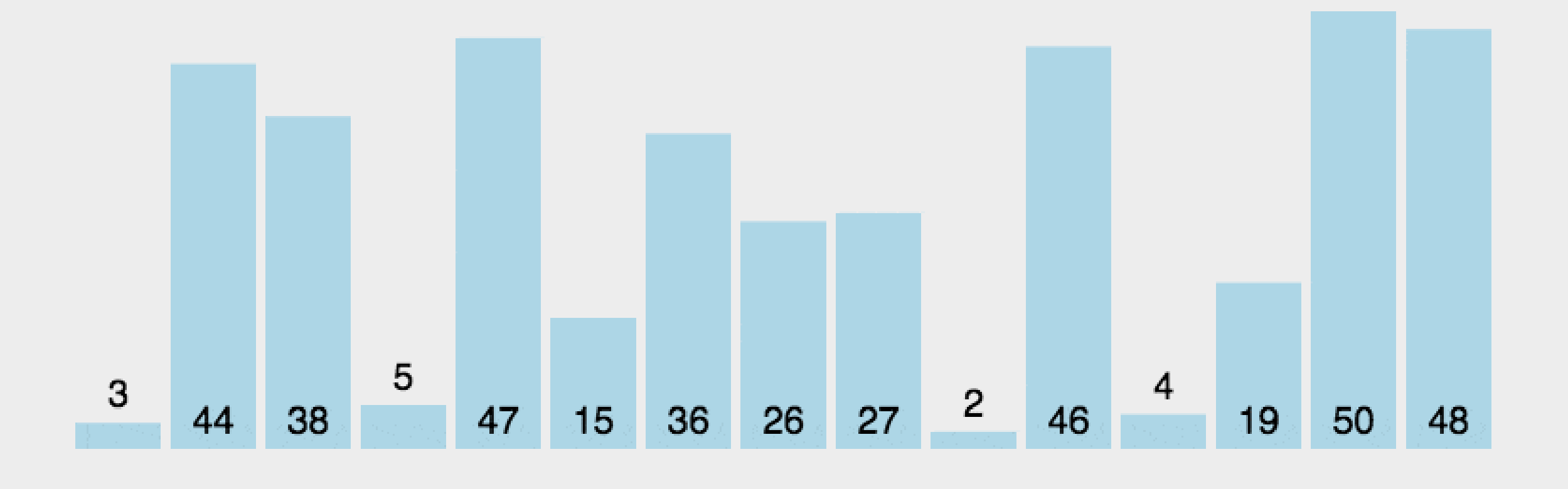
2	3	4	5	15	19	26	27	36	38	44	46	47	48	50

Operations of 1-D array: sorting



Bubble sort

In an array, compare the element with its neighbour and shift the larger/smaller one to one direction till end.



Bubble sort

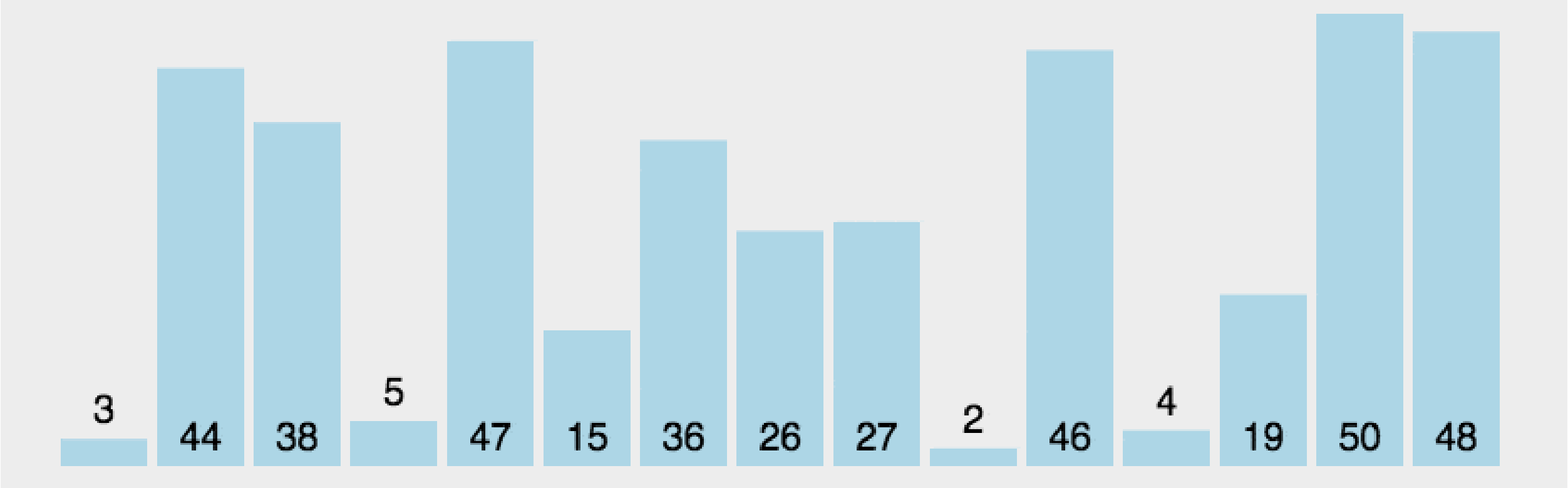
```c
#include<stdio.h>
int main(void) {
    int arr[] = { 22, 34, 3, 32, 82, 55, 89, 50, 37, 5, 64, 35, 9, 70 };
    int len = (int)sizeof(arr) / sizeof(arr[0]);

    for (int i = 0; i < len - 1; i++) // for each element
        for (int j = 0; j < len - 1 - i; j++) // compare with rest
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }

    for (int i = 0; i < len; i++)
        printf("%d \n", arr[i]);
}
```

```
C:\U...
22
34
3
32
82
55
89
50
37
5
64
35
9
70
```

```
Microso
3
5
9
22
32
34
35
37
50
55
64
70
82
89
```

Selection sort

In an array, find the max/min of the i to N elements and put at i-th location.



Selection sort

```c
#include<stdio.h>
int main(void) {
        int arr[] = { 22, 34, 3, 32, 82, 55, 89, 50, 37, 5, 64, 35, 9, 70 };
        int len = (int)sizeof(arr) / sizeof(*arr);
        for (int i = 0; i < len - 1; i++) {
                int min = i;
                for (int j = i + 1; j < len; j++) {
                        if (arr[j] < arr[min])
                                min = j;
                }
                int temp = arr[min];
                arr[min] = arr[i];
                arr[i] = temp;
        }
        int i;
        for (i = 0; i < len; i++)
                printf("%d\n", arr[i]);
}
```

C:\U...

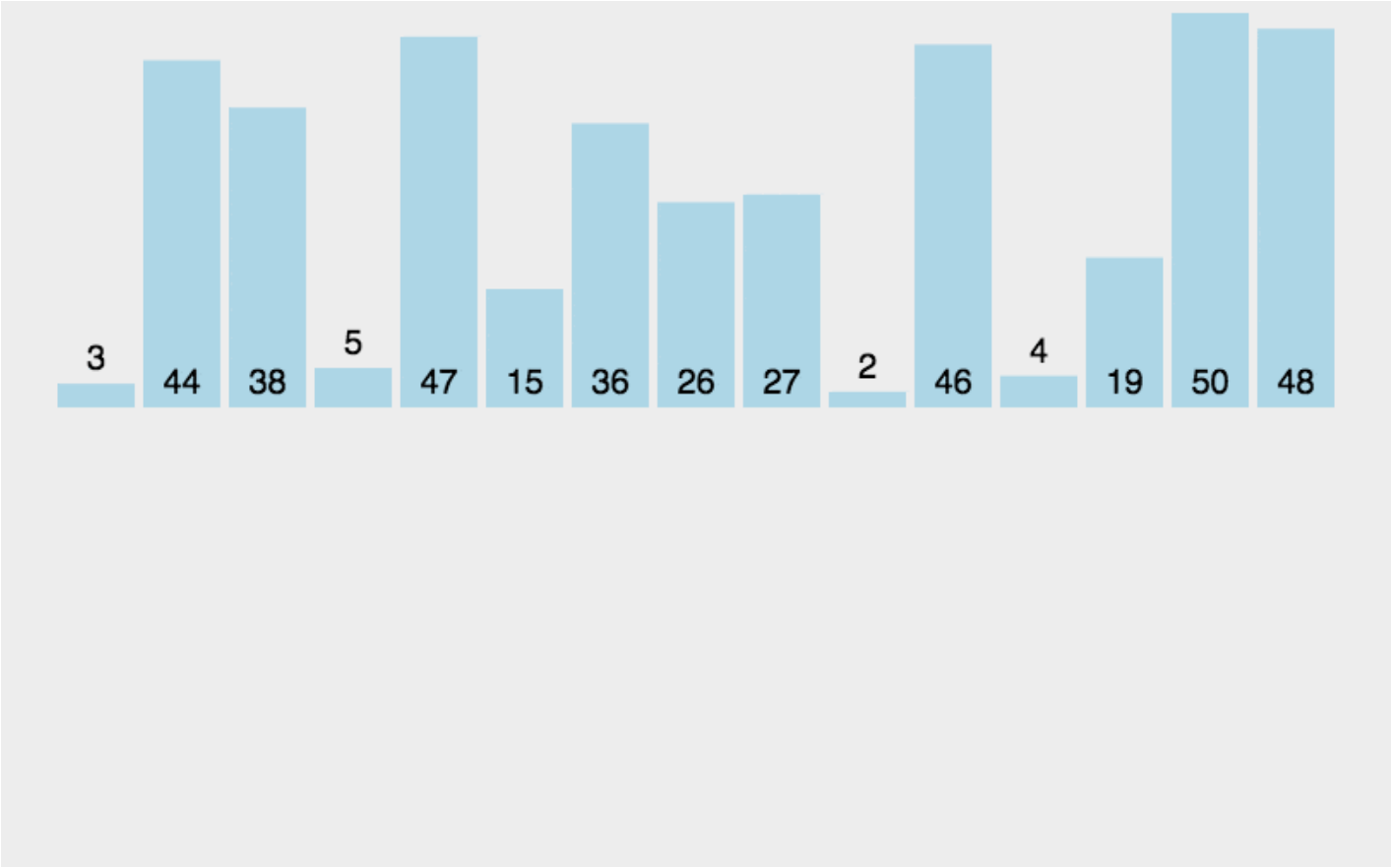
```
22
34
3
32
82
55
89
50
37
5
64
35
9
70
```

Microso

```
3
5
9
22
32
34
35
37
50
55
64
70
82
89
```

Insertion sort

In an array, compare the i-th element with its precedents and put it at the larger/smaller location.



Insertion sort

?

Try it yourself!

Content

1. 1-D array
2. 2-D and N-D array
3. String
4. Row-major or column-major order

1-D array to 2-D array

1-D array

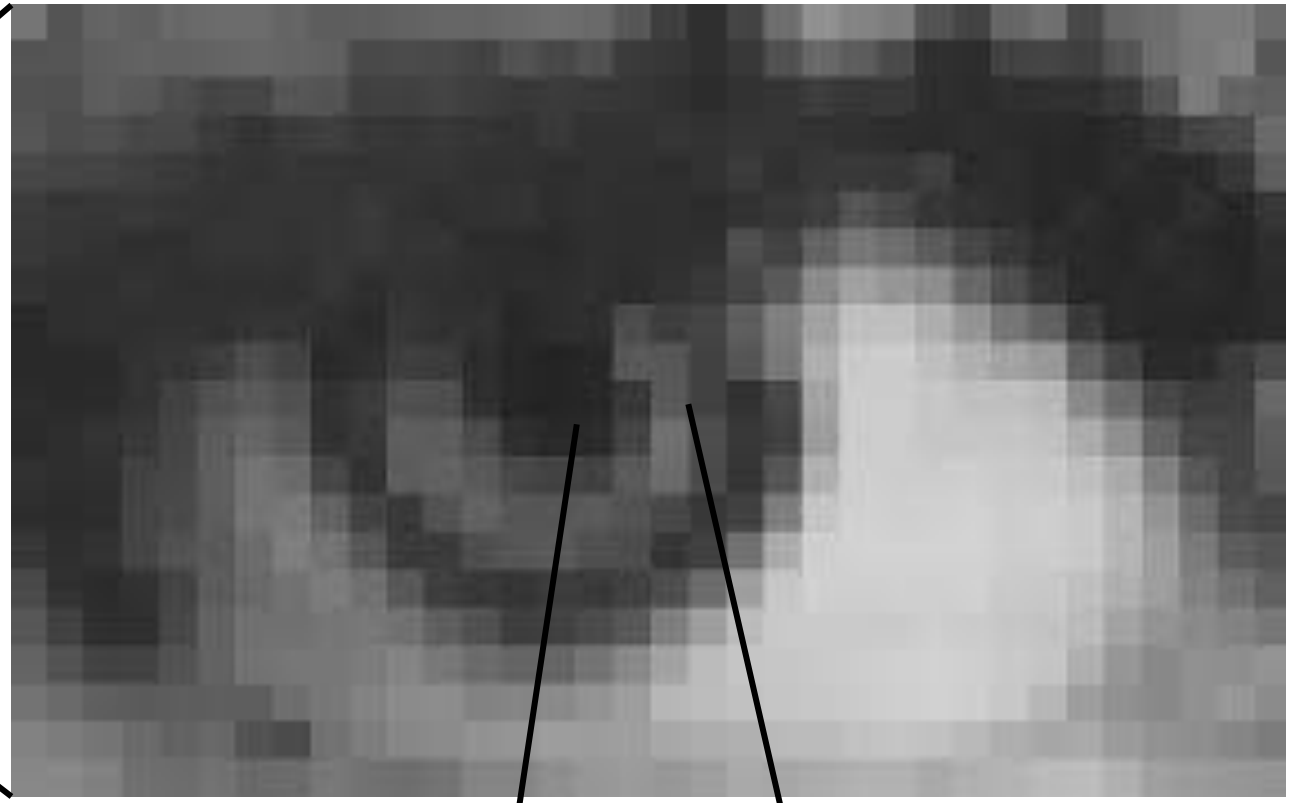
2-D array

2-D array in life

2-D array in life

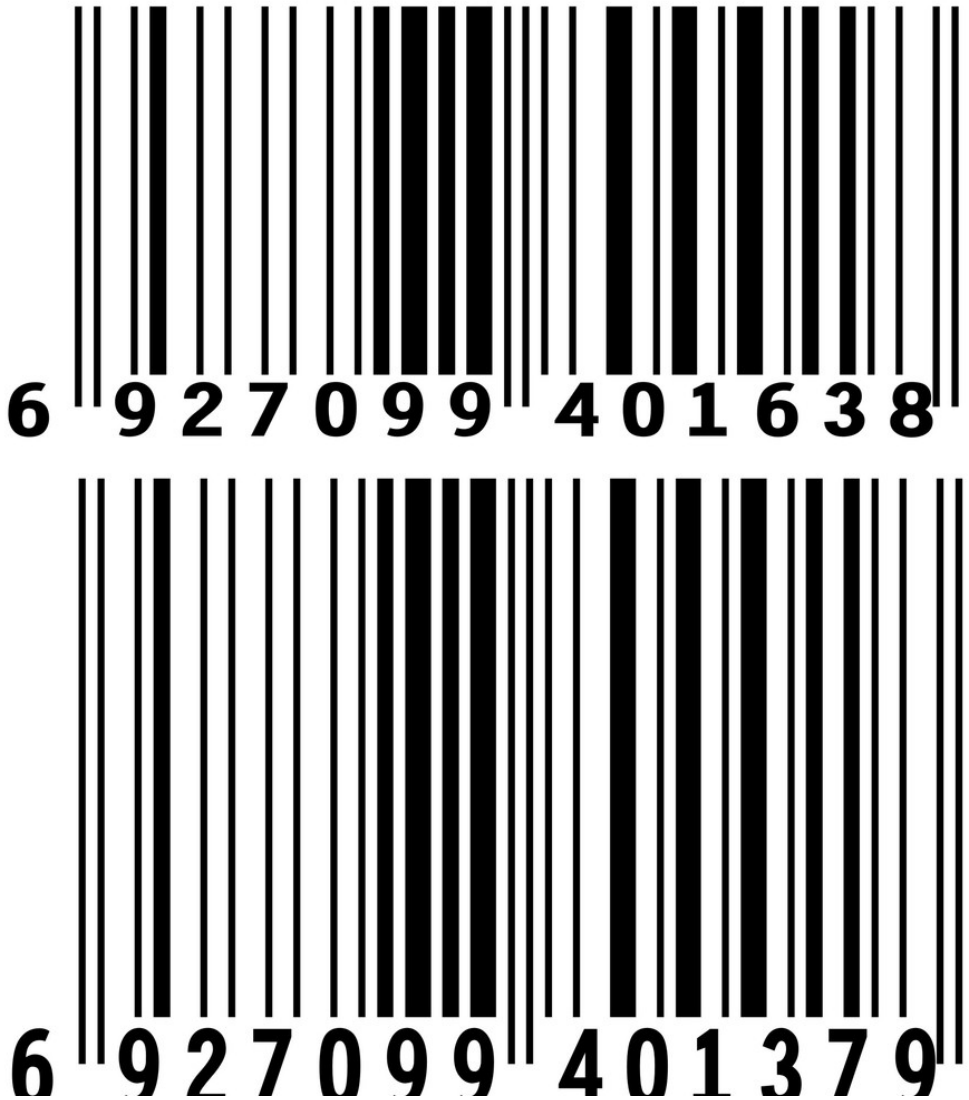
2-D array in life

Matrix

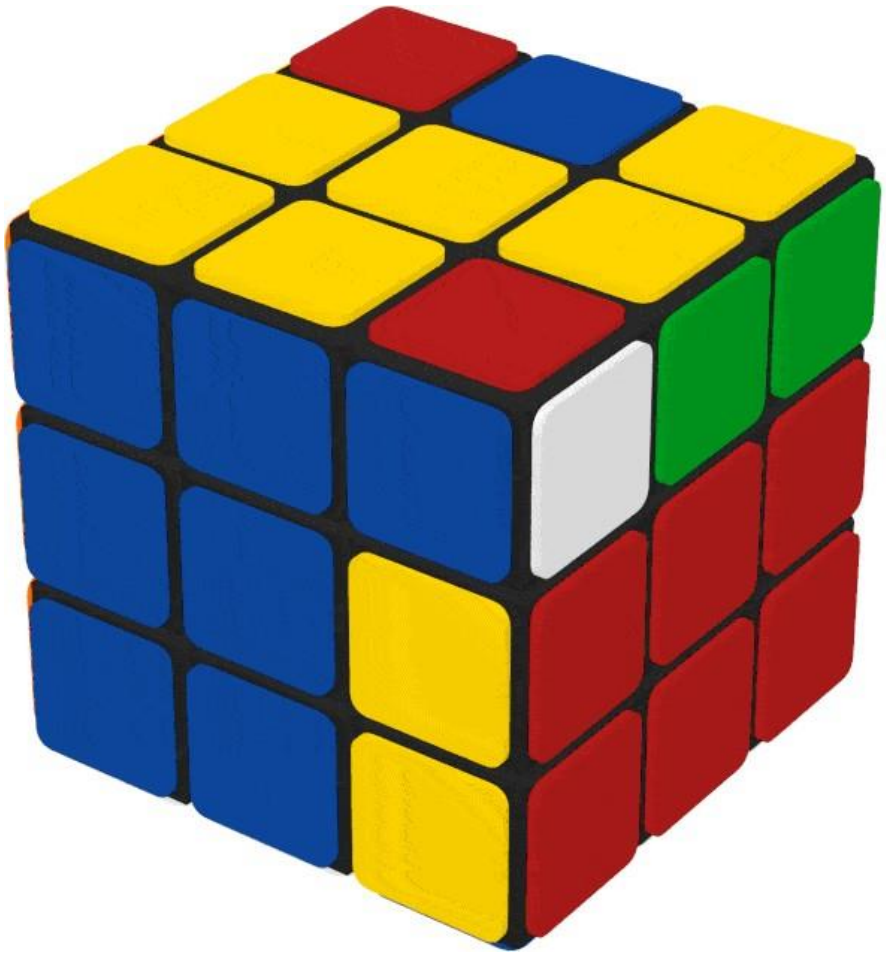


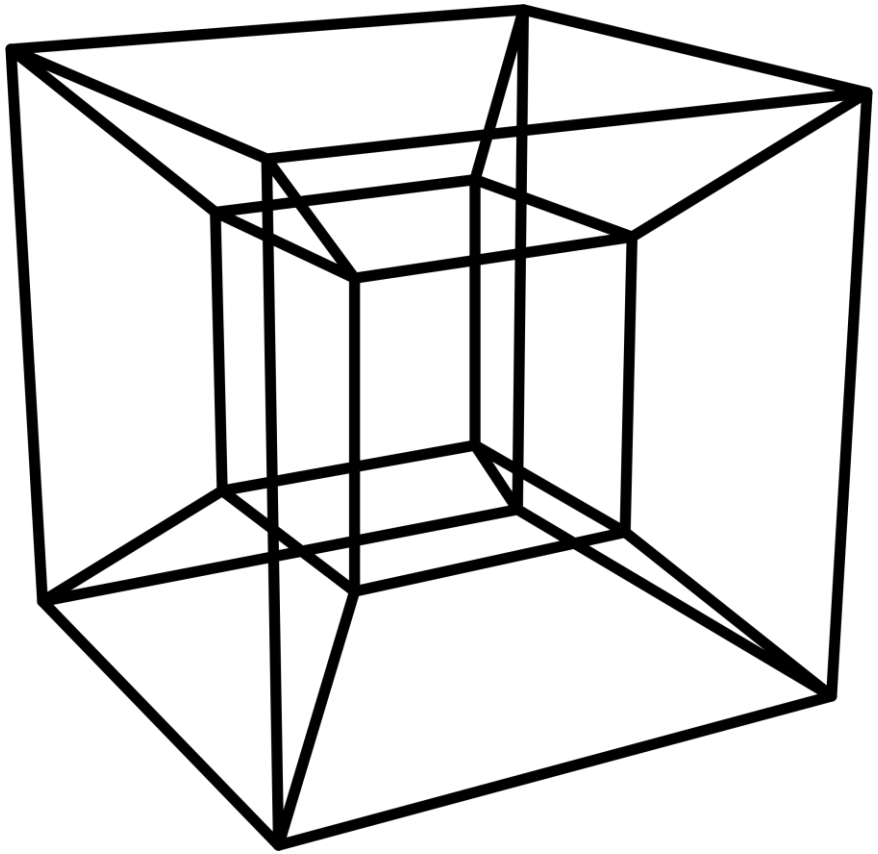
10	20	20	26
20	41	10	80
60	22	16	84

2-D array in life

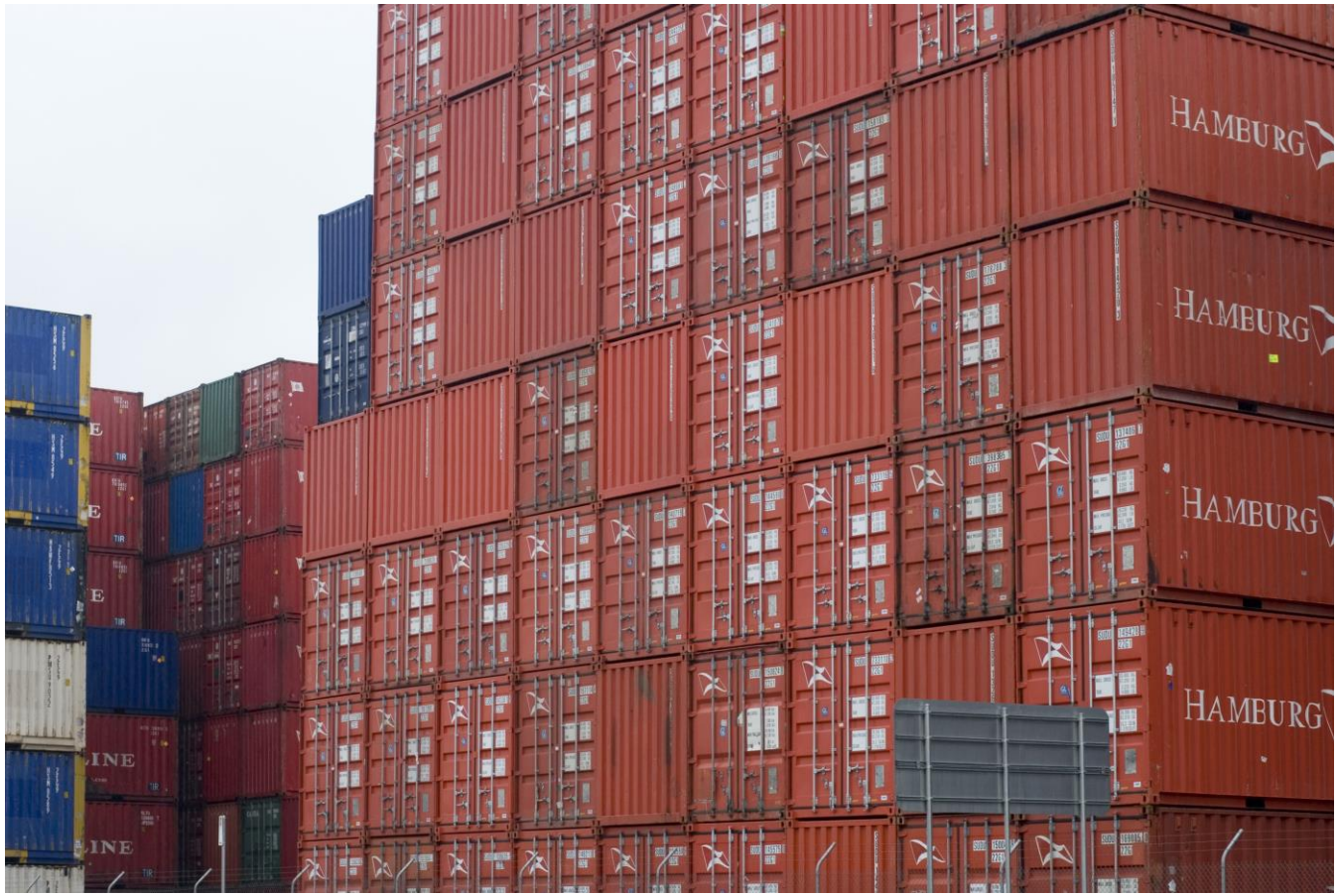


3-D/N-D array in life





3-D/N-D array in life



2-D array

1D-array can be extended to **2D structure**, with (X, Y) indexing the element.

```
type name[size][size];

type name[size][size] = {{…}, {…},…, {…}};

type name[][] = {{…}, {…},…, {…}};
```

2-D array

Declare and initialize a 2D int array

3	2	5
1	7	6

- int a[2][3]; ***// 2 rows x 3 columns***
- a[0][0] = 3; a[0][1] = 2; a[0][2] = 5;
- a[1][0] = 1; a[1][1] = 7; a[1][2] = 6;

Access array: printf("a[1][1] = %d", a[1][1]);

1	0	0	2
0	1	0	0
0	2	1	4

- int a[3][4]; ***// 3 rows x 4 columns***
- a[0][0] = 1; a[0][1] = 0; a[0][2] = 0; a[0][3] = 2;
- a[1][0] = 0; a[1][1] = 1; a[1][2] = 0; a[1][3] = 0;
- a[2][0] = 0; a[2][1] = 2; a[2][2] = 1; a[2][3] = 4;

Access array: printf("a[2][3] = %d", a[2][3]);

2-D array

Declare and initialize a 2D int array

- int a[2][3] = {{1, 2, 3}, {4, 5, 6}};
- int a[2][3] = {1, 2, 3, 4, 5, 6}; // **preferred!**
- int a[][3] = {1, 2, 3, 4, 5, 6}; // 2 x 3 mat
- int a[3][4] ={{1}, {5, 6}}; // 3 x 4 mat

1	2	3
4	5	6

1	0	0	0
5	6	0	0
0	0	0	0

Initialize

1．分行给二维数组赋初值。

```
int a[3][4]={{1，2，3，4}，{5，6，7，8}，{9，10，11，12}};
```

2．可以将所有数据写在一个花括号内，按数组排列的顺序对各元素赋初值。

```
int a [3] [4] ={1，2，3，4，5，6，7，8，9，10，11，12};
```

Initialize

3 . 可以对部分元素赋初值。

例如： int a［3］［4］={{1}，{5}，{9}};

也可以对各行中的某一元素赋初值，如

int a［3］［4］={{1}，{0，6}，{0，0，0，11}};

1	0	0	0
5	6	0	0
0	**0**	**0**	**0**

1	0	0	0
0	6	0	0
0	0	0	11

1	0	0	0
5	0	0	0
9	**0**	**0**	**0**

也可以只对某几行元素赋初值。如：

int a［3］［4］={{1}，{5，6}};

Initialize

4．如果对全部元素都赋初值，则定义数组时对第一维的长度可以不指定，但第二维的长度不能省。

```
int a[3][4]={1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
```

等价于

```
int a[][4]={1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
```

在定义时也可以只对部分元素赋初值而省略第一维的长度，但应分行赋初值。

例如：`int a[][4]={{0, 0, 3}, {}, {0, 10}};`

0	0	3	0
0	0	0	0
0	10	0	0

2-D array

1. 下标可以是整型表达式:

a［2-1］［2*2-1］ √

但是，不要写成 a［2，3］，a［2-1，2*2-1］！！！！

2. 数组元素可以出现在表达式中，也可以被赋值

b［1］［2］=a［2］［3］/2

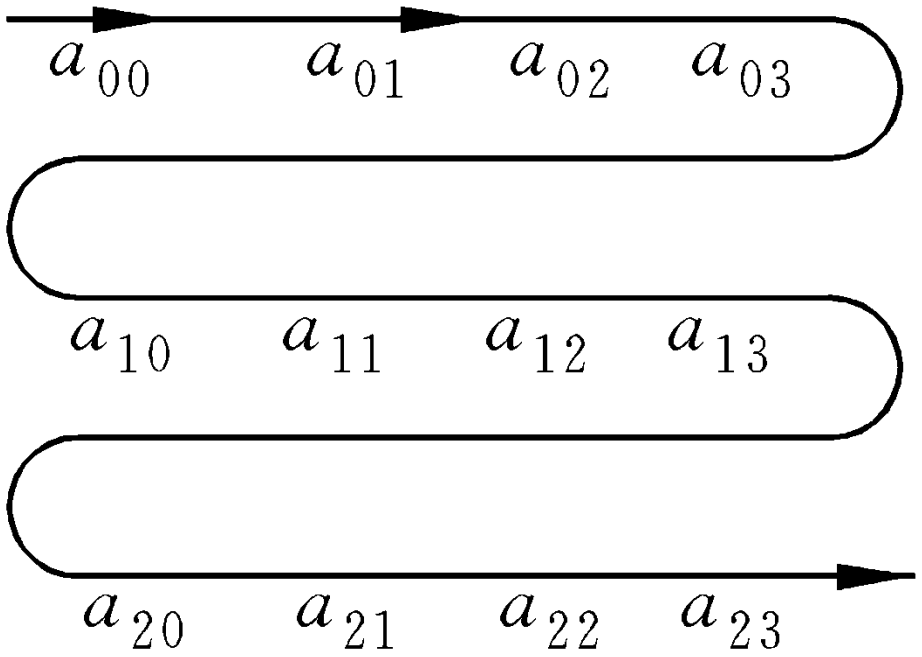
2-D array

3. 在使用数组元素时，应该注意下标值应在已定义的数组大小的范围内。

```
int a［3］［4］;   /* 定义a为3×4的数组 */
        ⋮
a［3］［4］=3;
```

2-D array

二维数组中的元素在内存中的排列顺序是：按行存放，即先顺序存放第一行的元素，再存放第二行的元素……



2-D array

整型数组 b[3][3]={ {1,2,3}, {4,5,6}, {7,8,9} };

地址	值	数组元素
3000H	1	b[0][0]
3002H	2	b[0][1]
3004H	3	b[0][2]
3006H	4	b[1][0]
3008H	5	b[1][1]
300AH	6	b[1][2]
300CH	7	b[2][0]
300EH	8	b[2][1]
3010H	9	b[2][2]

3-D/N-D array

Declare and initialize a 3-D/N-D int array

- int a[2][3][4];
- a[0][0][0] = 1; a[0][1][2] = 3; a[1][0][3] = 2; // **preferred!**
- int a[2][3][4]= {{{1, 2, 3}, {4, 5, 6}},{{2, 4, 5}, {2, 4, 2}}…};

- int a[2][3][4][2];
- a[0][0][0][0] = 1; a[0][1][2][0] = 3; a[1][0][3][1] = 2;

3-D/N-D array

定义三维数组： float a [2] [3] [4] ;

注意： 多维数组元素在内存中的排列顺序：

第一维的下标变化最慢，最右边的下标变化最快。

a[0][0][0]→a[0][0][1]→a[0][0][2]→a[0][0][3]→
a[0][1][0]→a[0][1][1]→a[0][1][2]→a[0][1][3]→
a[0][2][0]→a[0][2][1]→a[0][2][2]→a[0][2][3]→
a[1][0][0]→a[1][0][1]→a[1][0][2]→a[1][0][3]→
a[1][1][0]→a[1][1][1]→a[1][1][2]→a[1][1][3]→
a[1][2][0]→a[1][2][1]→a[1][2][2]→a[1][2][3]→

Use for loop to define 2D/3D array

2D array

```
int n[4][5];
for (int x = 0; x < 4; x++)
{
    for (int y = 0; y < 5; y++)
    {
        n[x][y] = x+y;
    }
}
```

3D array

```
int n[2][2][3];
for (int x = 0; x < 2; x++)
{
    for (int y = 0; y < 2; y++)
    {
        for (int z = 0; z < 3; z++)
        {
            n[x][y][z] = x+y+z;
        }
    }
}
```

Case study: 2-D array

Case: how to print a 2D float array and char array

```c
#include <stdio.h>
int main ()
{
float a[5][2] = { {0.5,1.5},
{1.2,2.1},{2.4,4.2},
{3.4,6.4},{4.4,8.5}};

for ( int i = 0; i < 5; i++ )
{
  for ( int j = 0; j < 2; j++ )
  {
    printf("%f ", a[i][j] );
  }
  printf("\n");
}
return 0;
}
```

```
0.500000 1.500000
1.200000 2.100000
2.400000 4.200000
3.400000 6.400000
4.400000 8.500000
```

```c
#include <stdio.h>
int main()
{
char a[5][2] = { {'A','B'}, {'C','D'},
{'E','F'}, {'G','H'},{'I','J'}};

for (int i = 0; i < 5; i++)
    {
        for (int j = 0; j < 2; j++)
        {
            printf("%c", a[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

```
AB
CD
EF
GH
IJ
```

Operations of 2-D array: matrix

Definition of matrix: **A matrix is a collection of numbers arranged into a fixed number of rows and columns.** 

$$\begin{pmatrix} 2 & 5 & 4 \\ 1 & 3 & 6 \\ 7 & 2 & 3 \end{pmatrix}$$

Operations of 2-D array: matrix

Most decisions can be expressed as a linear equation!

$$a \cdot X_1 + b \cdot X_2 + c \cdot X_3 = Y$$

a, b, c are system coefficients

X is observation or measurement

Y is decision

Operations of 2-D array: matrix

$$a \cdot X_1 + b \cdot X_2 + c \cdot X_3 = Y$$

$$\begin{pmatrix} a & b & c \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = Y$$

system coefficients

observation or measurement

decision

Operations of 2-D array: matrix

Multiple measurements build up a matrix

$$\begin{pmatrix} a & b & c \end{pmatrix} \cdot \begin{pmatrix} X_{11} & X_{21} & X_{31} \\ X_{12} & X_{22} & X_{32} \\ X_{13} & X_{23} & X_{33} \end{pmatrix} = \begin{pmatrix} Y_1 & Y_2 & Y_3 \end{pmatrix}$$

system coefficients

observation or measurement

decision

Basic matrix operations

+, -, .*, ./, *, /

$$A = \begin{pmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \end{pmatrix} \qquad B = \begin{pmatrix} b_{11} & b_{21} \\ b_{12} & b_{22} \end{pmatrix}$$

Basic matrix operations

Matrix adding and subtraction

$$A \pm B = \begin{bmatrix} a_{00} \pm b_{00}, & a_{01} \pm b_{01}, & \cdots & a_{0j} \pm b_{0j} \\ a_{10} \pm b_{10}, & a_{11} \pm b_{11}, & \cdots & a_{1j} \pm b_{1j} \\ \cdots & \cdots & \cdots & \cdots \\ a_{i0} \pm b_{i0}, & a_{i1} \pm b_{i1}, & \cdots & a_{ij} \pm b_{ij} \end{bmatrix}$$

Matrix dot product

$$A \cdot B = \begin{bmatrix} a_{00} \cdot b_{00}, & a_{01} \cdot b_{01}, & \cdots & a_{0j} \cdot b_{0j} \\ a_{10} \cdot b_{10}, & a_{11} \cdot b_{11}, & \cdots & a_{1j} \cdot b_{1j} \\ \cdots & \cdots & \cdots & \cdots \\ a_{i0} \cdot b_{i0}, & a_{i1} \cdot b_{i1}, & \cdots & a_{ij} \cdot b_{ij} \end{bmatrix}$$

Matrix cross product

$$A_{23} \times B_{32} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \times \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \\ b_{20} & b_{21} \end{bmatrix}$$

$$= \begin{bmatrix} a_{00} \cdot b_{00} + a_{01} \cdot b_{10} + a_{02} \cdot b_{20}, & a_{00} \cdot b_{01} + a_{01} \cdot b_{11} + a_{02} \cdot b_{21} \\ a_{10} \cdot b_{00} + a_{11} \cdot b_{10} + a_{12} \cdot b_{20}, & a_{10} \cdot b_{01} + a_{11} \cdot b_{11} + a_{12} \cdot b_{21} \end{bmatrix}$$

Basic matrix operations

How to transpose a matrix?

int a[2][3] = {{1, 2, 3}, {4, 5, 6}};

1	2	3
4	5	6

Rotate 90°

int a[3][2] = {{1, 4}, {2, 5}, {3, 6}};

1	4
2	5
3	6

Case study: 2-D array

Case: how to transpose a 2D matrix?

```
#include <stdio.h>
main()
{
    int a[2][3] = {{1, 2, 4}, {4, 5, 2}};
    int a_trans[3][2];
    for (int i = 0; i < 2; i++){
        for (int j = 0; j < 3; j++){
            a_trans[j][i] = a[i][j];
        }
    }
    printf("\nMatrix A:\n");
    for (int i = 0; i < 2; i++){
        for (int j = 0;j < 3; j++){
           printf("%d ", a[i][j]);}
        printf("\n");}
    printf("\nTranspose of matrix A:\n");
    for (int i = 0; i < 3; i++){
        for (int j = 0; j < 2; j++){
            printf("%d ", a_trans[i][j]);}
        printf("\n");}
}
```

```
Matrix A:
1 2 4
4 5 2

Transpose of matrix A:
1 4
2 5
4 2
```

Basic matrix operations

How to turn a matrix upside down?

3	6
2	5
1	4

upside down

1	4
2	5
3	6

Case study: 2-D array

Case: how to turn a 2D matrix upside down?

```
#include <stdio.h>
int main(){
    int a[][2] = {1, 2, 3, 4, 5, 6};
    int i, j, m, n, tmp;
    m = 3; n = 2;
    printf("Original matrix:\n");
    for(i = 0; i < m; i ++)
        for(j = 0; j < n; j ++)
            if(j == 1) printf("%d\n", a[i][j]);
            else printf("%d ", a[i][j]);
    for(i = 0; i < m/2; i ++)
        for(j = 0; j < n; j ++){
            tmp = a[i][j]; a[i][j] = a[m-1-i][j]; a[m-1-i][j] = tmp;
        }
    printf("Upside down matrix:\n");
    for(i = 0; i < 3; i ++)
        for(j = 0; j < 2; j ++)
            if(j == 1) printf("%d\n", a[i][j]);
            else printf("%d ", a[i][j]);
    return 0;
}
```

```
Original matrix:
1 2
3 4
5 6
Upside down matrix:
5 6
3 4
1 2
```

Case study: 2-D array

Case: how to reverse the left and right?

```
#include <stdio.h>
int main(){
    int a[][2] = {1, 2, 3, 4, 5, 6};
    int i, j, m, n, tmp;
    m = 3; n = 2;
    printf("Original matrix:\n");
    for(i = 0; i < m; i ++)
        for(j = 0; j < n; j ++)
            if(j == 1) printf("%d\n", a[i][j]);
            else printf("%d ", a[i][j]);
    for(i = 0; i < m; i ++)
        for(j = 0; j < n/2; j ++){
            tmp = a[i][j]; a[i][j] = a[i][n-1-j]; a[i][n-1-j] = tmp;
        }

    printf("Upside down matrix:\n");
    for(i = 0; i < 3; i ++)
        for(j = 0; j < 2; j ++)
            if(j == 1) printf("%d\n", a[i][j]);
            else printf("%d ", a[i][j]);
    return 0;
}
```

```
Original matrix:
1 2
3 4
5 6
Upside down matrix:
2 1
4 3
6 5
```

Case study: subtract 2 matrices

Case: how to subtract 2 matrices?

```
#include <stdio.h>
main()
{
    int a[2][2] = {{1, 2},{4, 5}};
    int b[2][2] = {{2, 2},{1, 3}};
    int c[2][2];
    for (int i = 0; i < 2;i++){
        for (int j = 0; j < 2; j++){
            c[i][j] = a[i][j] - b[i][j];
        }
    }
    printf("Matrix A-B:\n");
    for(int i = 0; i < 2; i++){
        for (int j = 0; j < 2; j++){
            printf("%d ", c[i][j]);
        }
        printf("\n");
    }
}
```

```
Matrix A:
1 2
4 5

Matrix B:
2 2
1 3
```

```
Matrix A-B:
-1 0
3 2
```

Case study: dot multiplication

Case: how to dot multiply 2 matrices?

```
#include <stdio.h>
main()
{
    int a[2][2] = {{1, 2},{4, 5}};
    int b[2][2] = {{2, 2},{1, 3}};
    int c[2][2];
    for (int i = 0; i < 2;i++){
        for (int j = 0; j < 2; j++){
            c[i][j] = a[i][j] * b[i][j];
        }
    }
    printf("Hadamard product of A and B:\n");
    for(int i = 0; i < 2; i++){
        for (int j = 0; j < 2; j++){
            printf("%d ", c[i][j]);
        }
        printf("\n");
    }
}
```

```
Matrix A:
1 2
4 5

Matrix B:
2 2
1 3
```

```
Hadamard product of A and B:
2 4
4 15
```

Case study: cross multiplication

Case: how to cross multiply 2 matrices?

```
#include <stdio.h>
main()
{
    int a[2][2] = {{1, 2},{4, 5}};
    int b[2][3] = {{2, 2, 1},{1, 3, 2}};
    int c[2][3];
    for (int i = 0; i < 2;i++){
        for (int j = 0; j < 3; j++){
            for (int k = 0; k < 2; k++){
                if(k==0)
                    c[i][j]=a[i][k]*b[k][j];
                else
                    c[i][j]+=a[i][k]*b[k][j];
        }}}
    printf("Cross product of A and B:\n");
    for(int i = 0; i < 2; i++){
        for (int j = 0; j < 3;j++){
            printf("%d ", c[i][j]);
        }
        printf("\n");
    }
}
```

```
Matrix A:
1 2
4 5

Matrix B:
2 2 1
1 3 2

Cross product of A and B:
4 8 5
13 23 14
```