Introduction to C Programming

Lecture 4: array & string

Wenjin Wang
wangwangj3@sustech.edu.cn

9-30-2022

Course syllabus

Nr.	Lecture	Date
1	Introduction	2022.9.9
2	Basics	2022.9.16
3	Decision and looping	2022.9.23
4	Array & string	2022.9.30
5	Functions	2022.10.9 (补)
6	Pointer	2022.10.14
7	Self-defined types	2022.10.21
8	File I/O & head files	2022.10.28

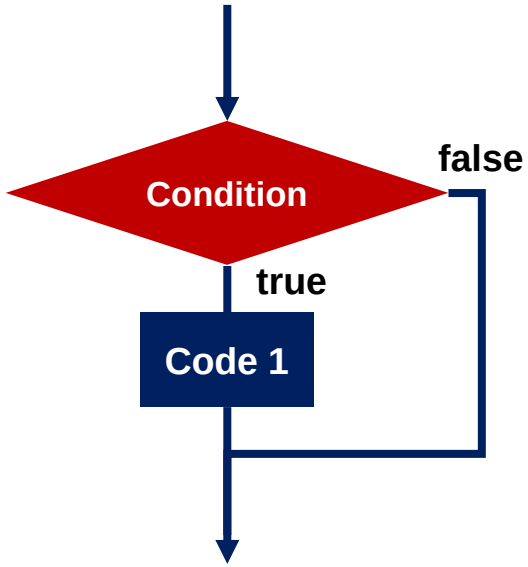
Nr.	Lecture	Date
9	Software engineering	2022.11.4
10	Review of lectures	2022.11.11
11	Soul of programming: Algorithms I	2022.11.25
12	Soul of programming: Algorithms II	2022.12.2
13	R&D project	2022.12.9
14	R&D project	2022.12.16
15	R&D project	2022.12.23
16	Summary	2023.12.30

Recap last lectures

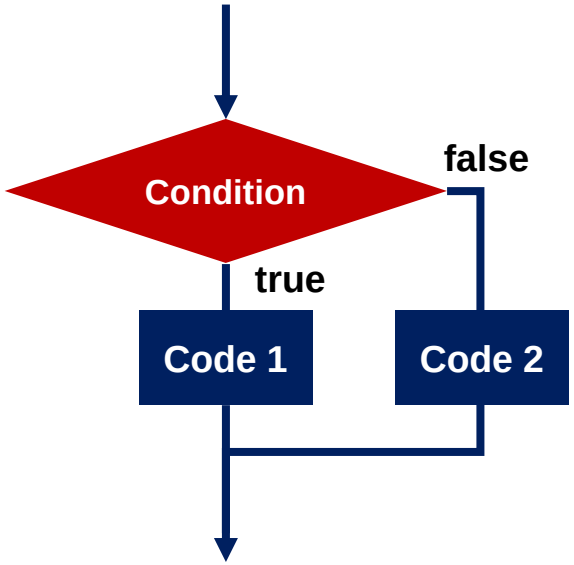
- Two types of workflow controls in C: **decision-making** and **looping**
- Decision making has **if-else** and **switch-case**, conditions can be nested
- Looping has **for-loop** and **while-loop**, loops can be nested
- You can control workflows in C

Recap last lectures

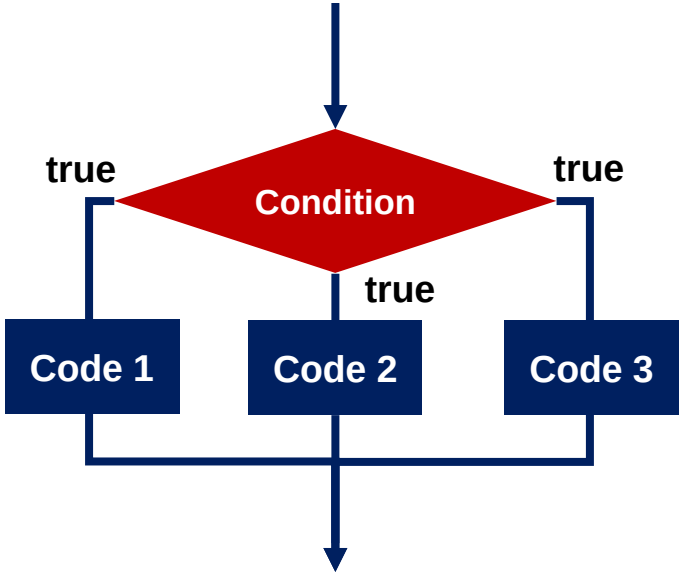
If



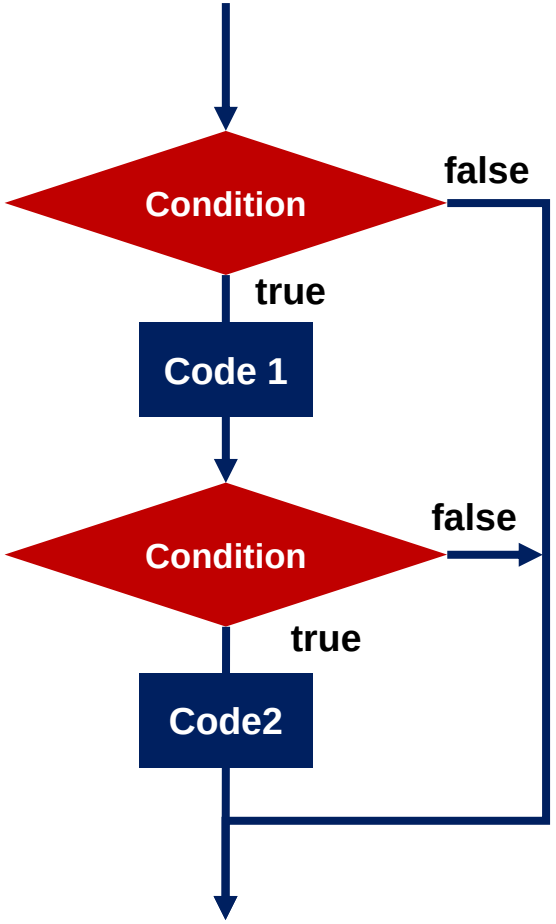
If else



If elseif



Nested if



Recap last lectures

If

```
int a = 3;

if (a > 10)
{
  // …
}
```

If else

```
int a = 3;

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

If elseif

```
int a = 3;

if (a > 10)
{
  // …
}else if(a > 5)
{
  // …
}
else
{
  // …
}
```

Nested if

```
int a = 3, b = 5;

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

Recap last lectures

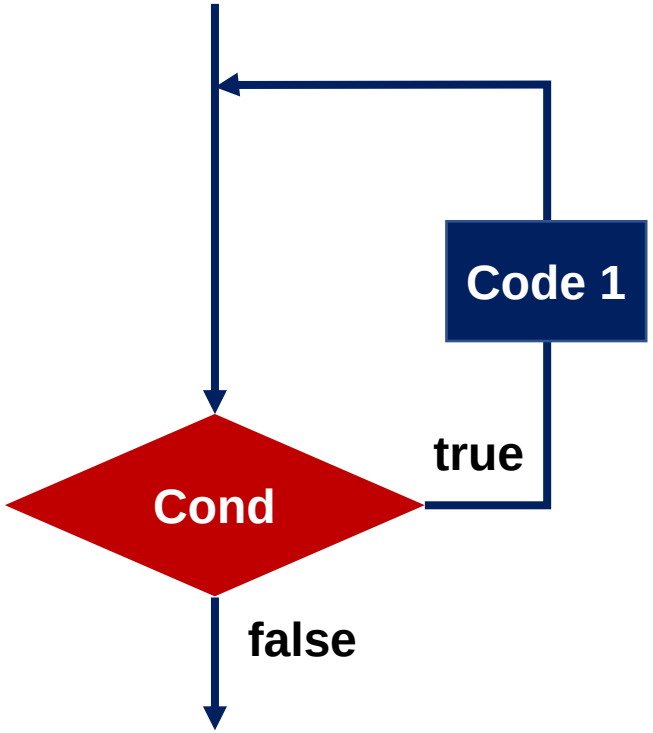
```
int a = 3;

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

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

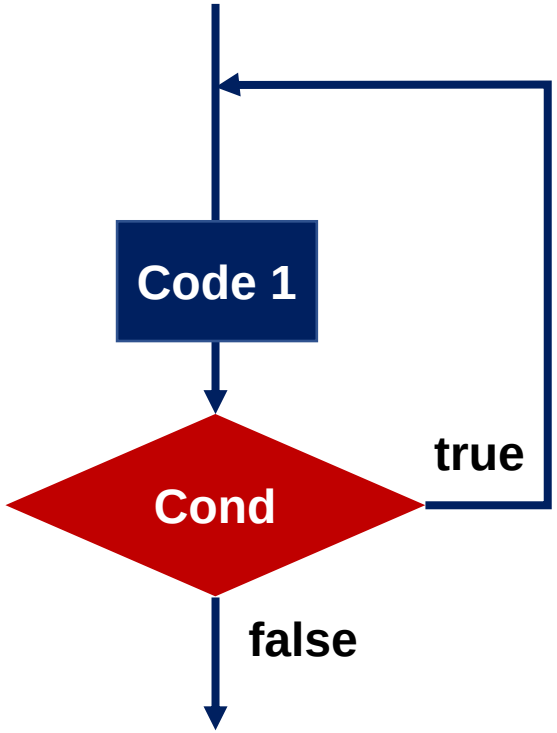
Switch can only express equality!!!

Recap last lectures

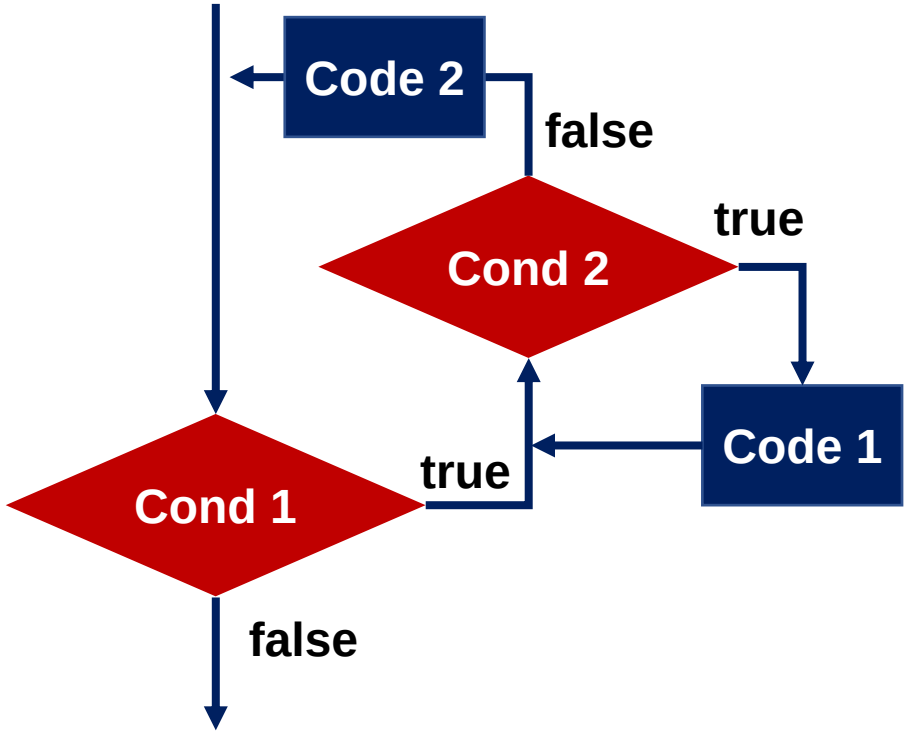
for/while loop



do-while loop

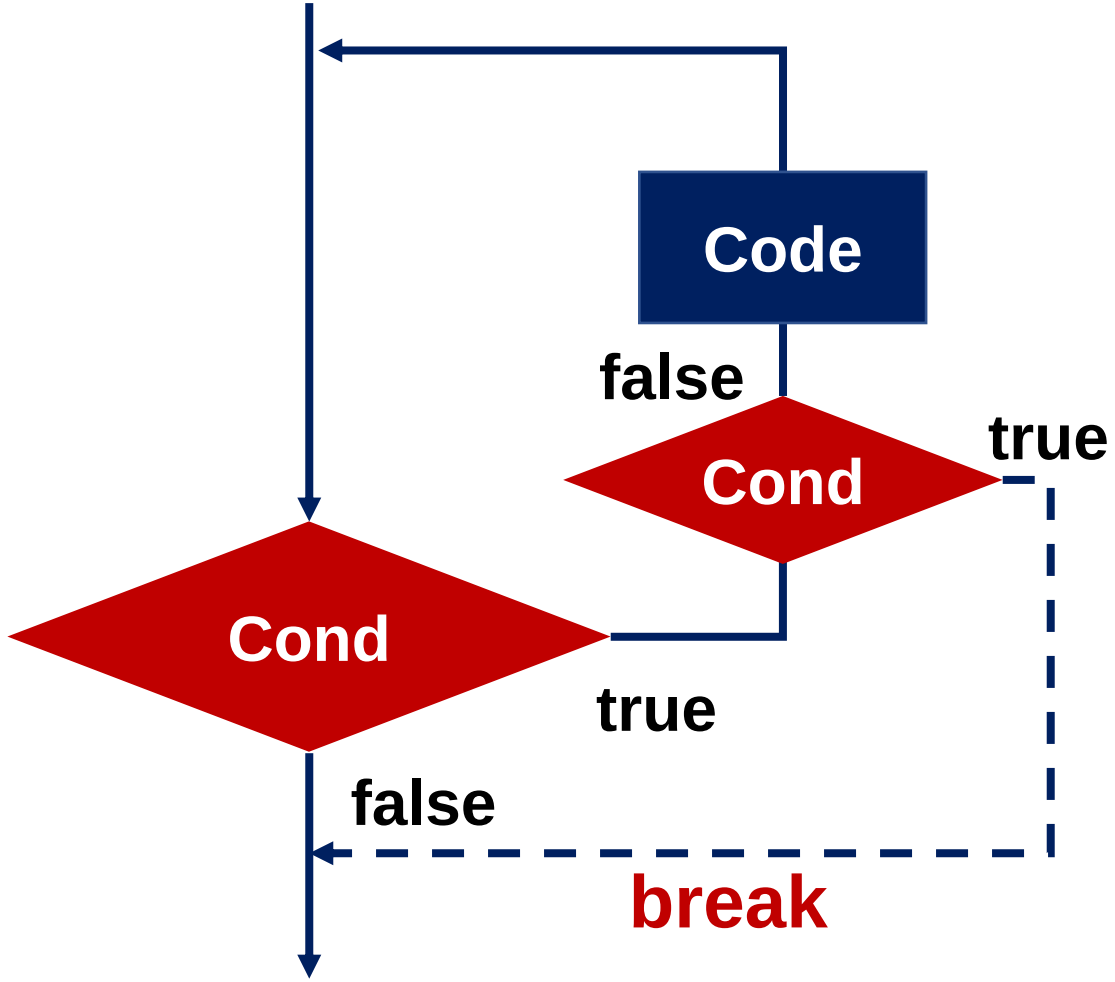


nested for loop

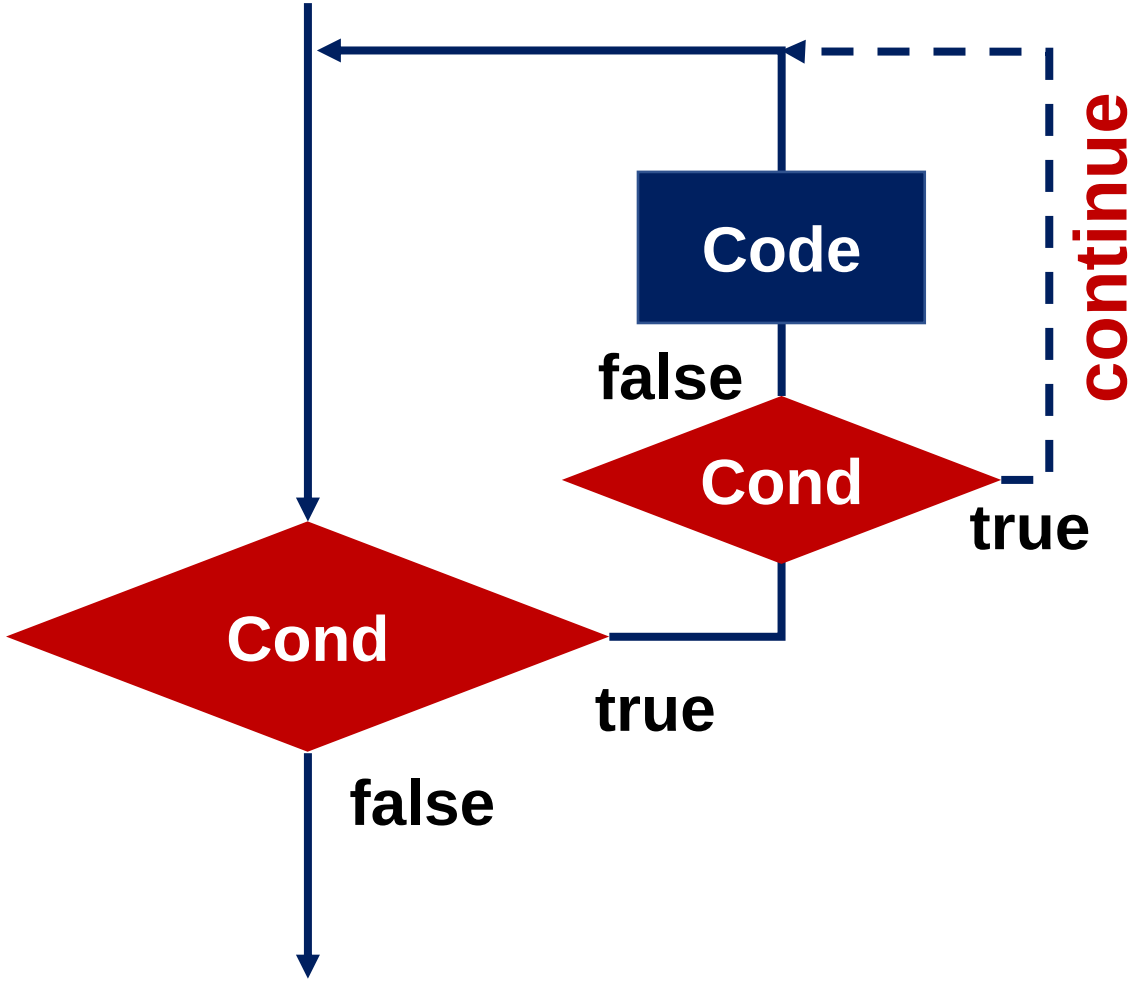


Recap last lectures

Break loop



Continue loop



Recap last lectures

for loop

```
for(int a = 0; a < 10; a++)
{
    // …
}
```

while loop

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

do-while loop

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

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

Content

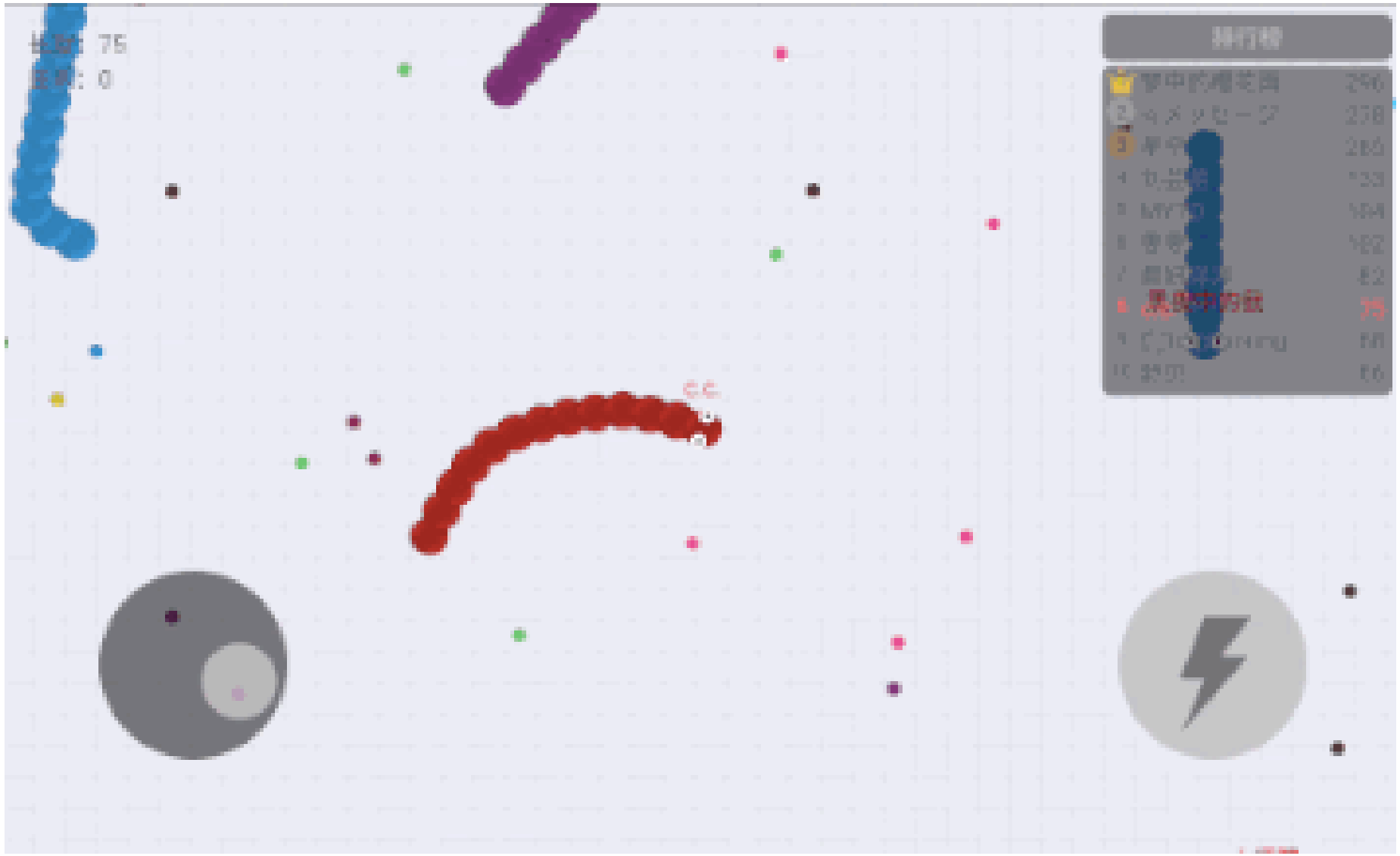
1. 1-D array
2. 2-D and N-D array
3. String

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);
    }
}
```

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

1-D array

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]**

Index starts at 0

Can we access array by a[10]?

1-D array

```
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

1-D array

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]

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

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

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’, ‘!’};

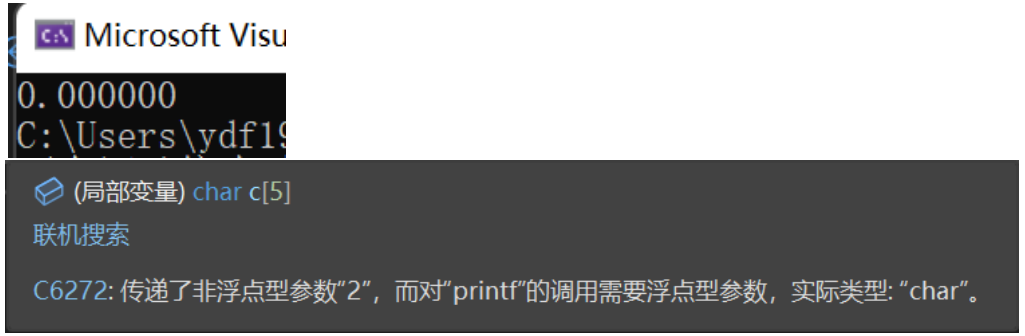
‘h’	‘e’	‘l’	‘l’	‘o’	‘!’

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!***

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

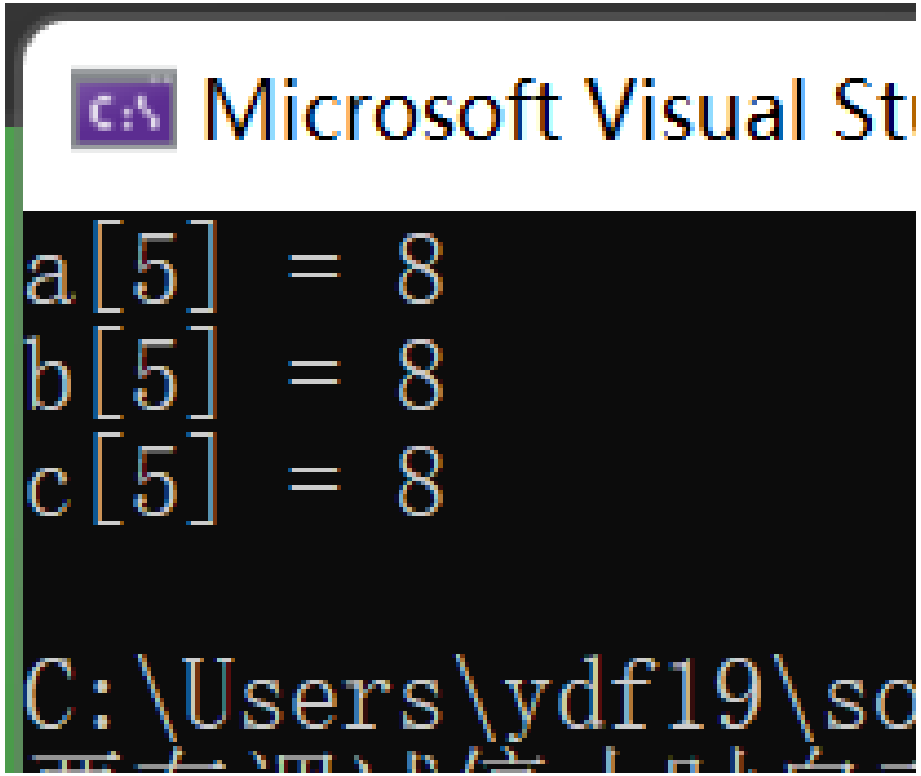



```
main()
{
    int c[5] = {0,1,2,2.5,5};
    printf("%f",c[3]);
}
```

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?

```
main()
{
    int carriages[6] = {34,56,89,32,76,39};
    int all = 0;
    for (int i = 0; i < 6; i++)
        all += carriages[i];

    printf("There are %d people on the train",
    all);
}
```

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]);
}
```

Microsoft Vi

```
SUSTech
SUSTech
C:\Users\ydf
```

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

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

Case: make four basic operations between two integer arrays.

Microsoft Visual Studio 调试控制台

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

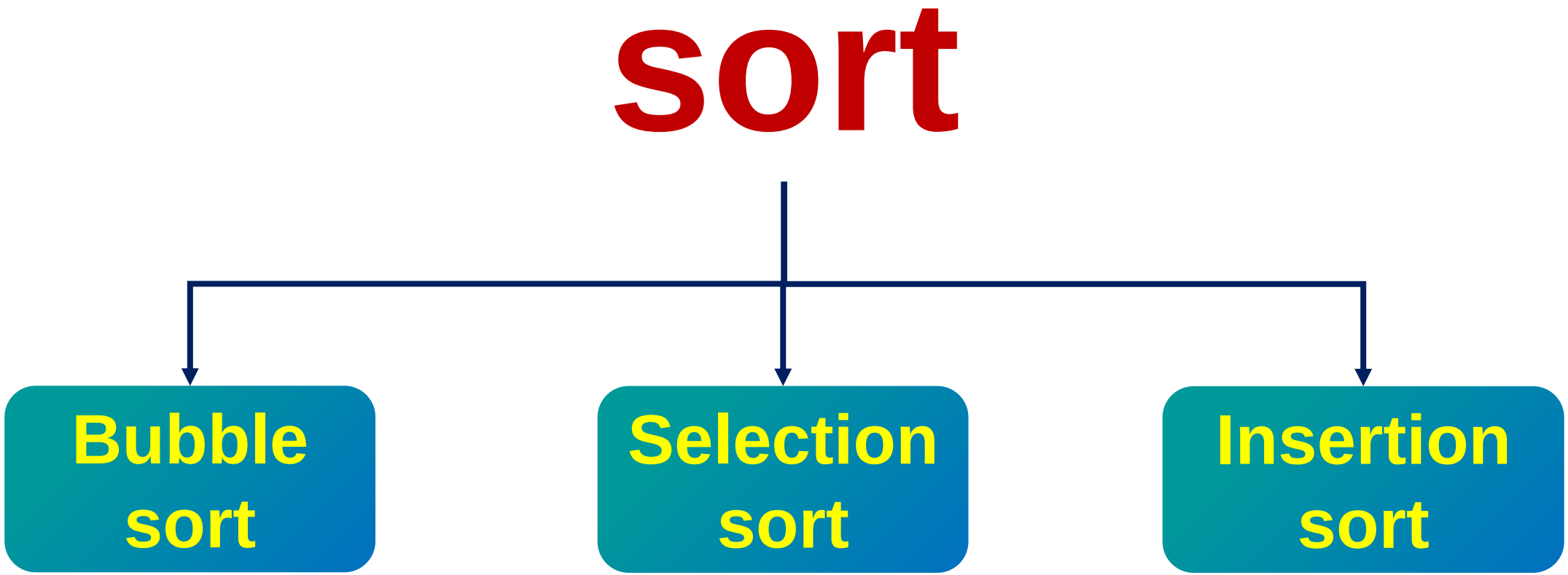
```
int a[10] = {3, 2, 1, 5, 6, 8, 9, 2, 0, 7};
```

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

How the sort the array?

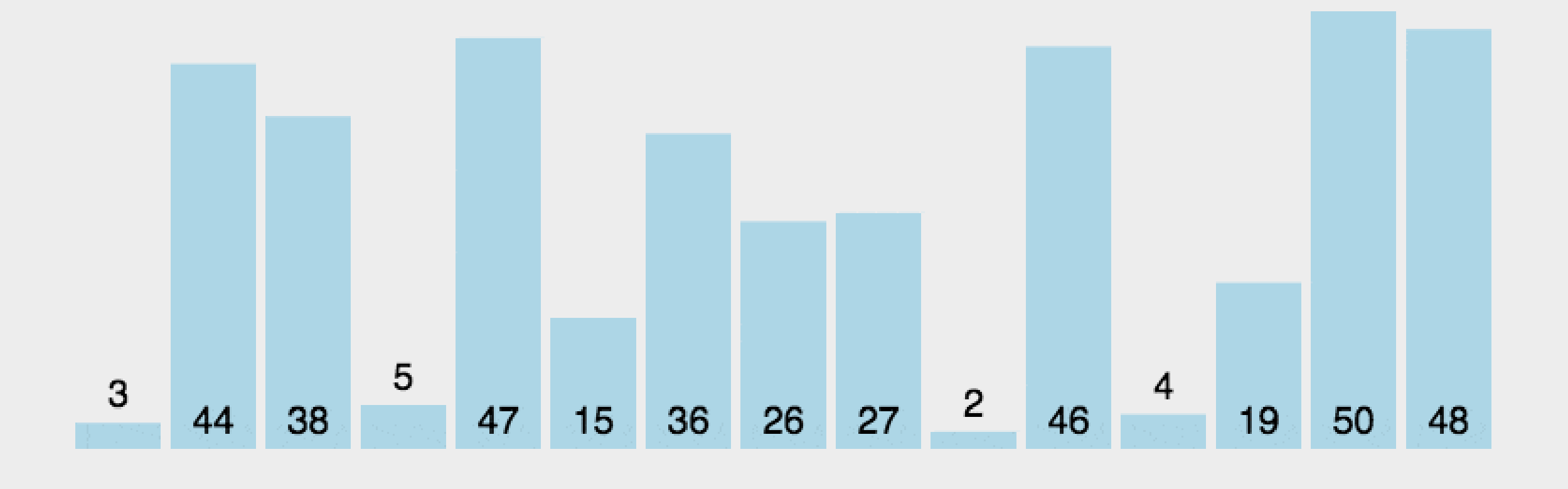
2	3	4	5	15	19	26	27	36	38	44	46	47	48	50

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

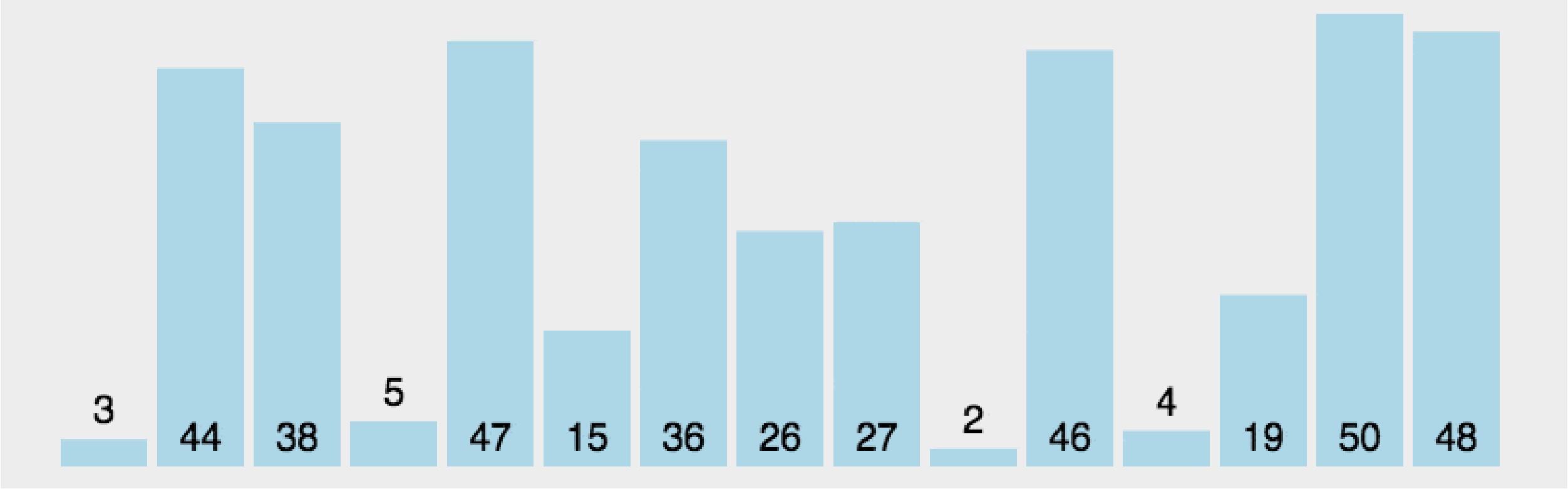
```c
main() {
    int arr[] = { 22, 34, 3, 32, 82, 55, 89, 50, 37, 5, 64, 35, 9, 70 };
    int len = (int)sizeof(arr) / sizeof(arr[0]);

    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]);
}
```

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

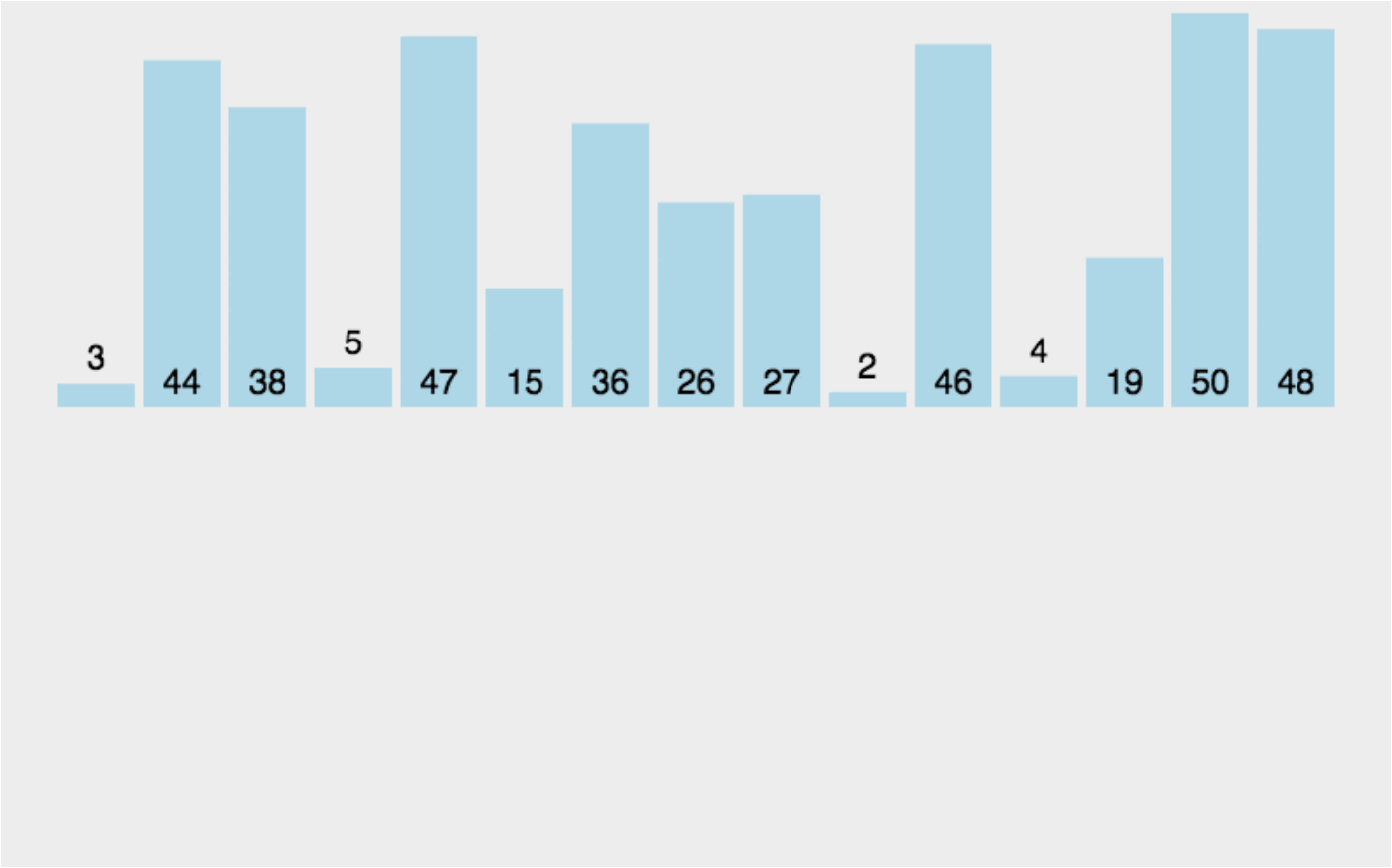
```c
main() {
    int arr[] = { 22, 34, 3, 32, 82, 55, 89, 50, 37, 5, 64, 35, 9, 70 };
    int len = (int)sizeof(arr) / sizeof(*arr);
    for (int i = 0; i < len - 1; i++)
    {
        int min = i;
        for (int j = i + 1; j < len; j++)
            if (arr[j] < arr[min])
                min = j;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

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.



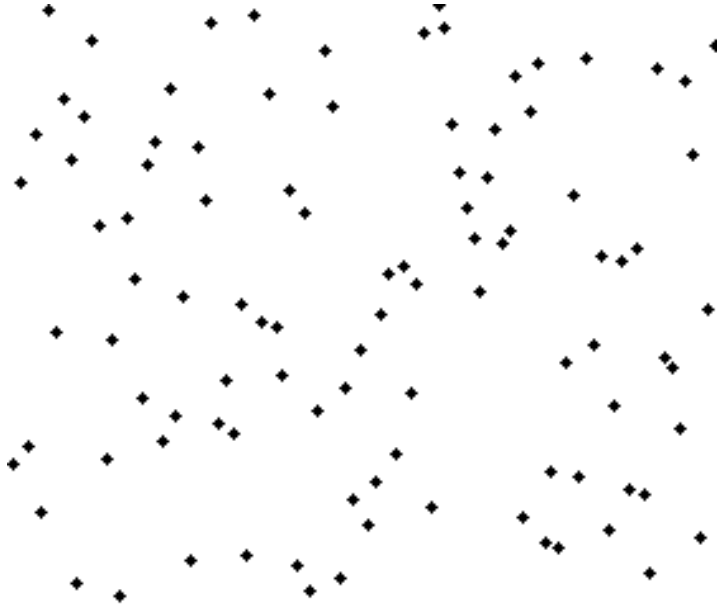
Insertion sort

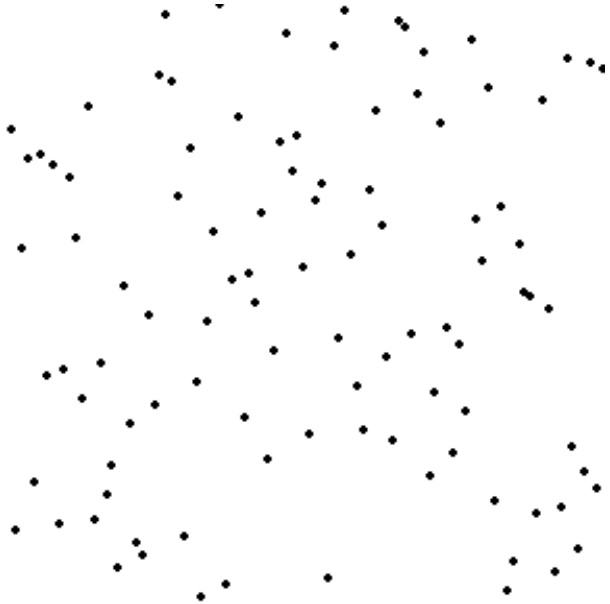
?

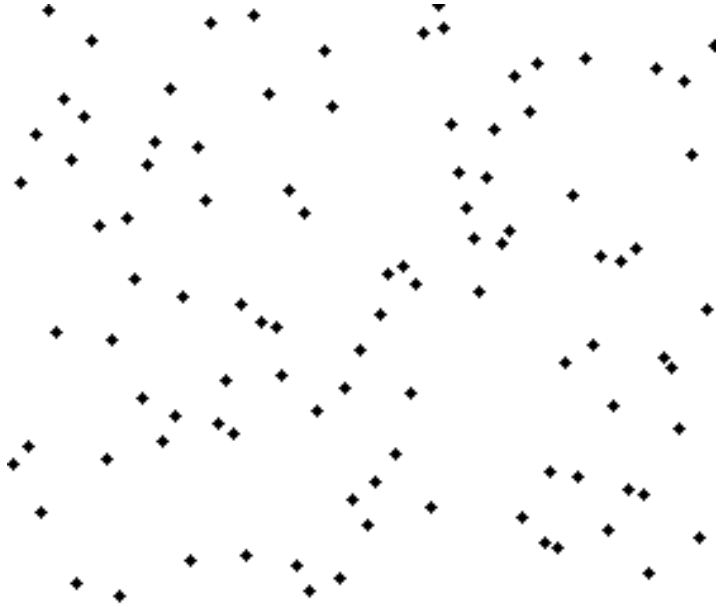
Try it yourself!

Overview of sorting

Which one is bubble, selection and insertion sorting?







Content

1. 1-D array
2. 2-D and N-D array
3. String

1-D array to 2-D array

1-D array

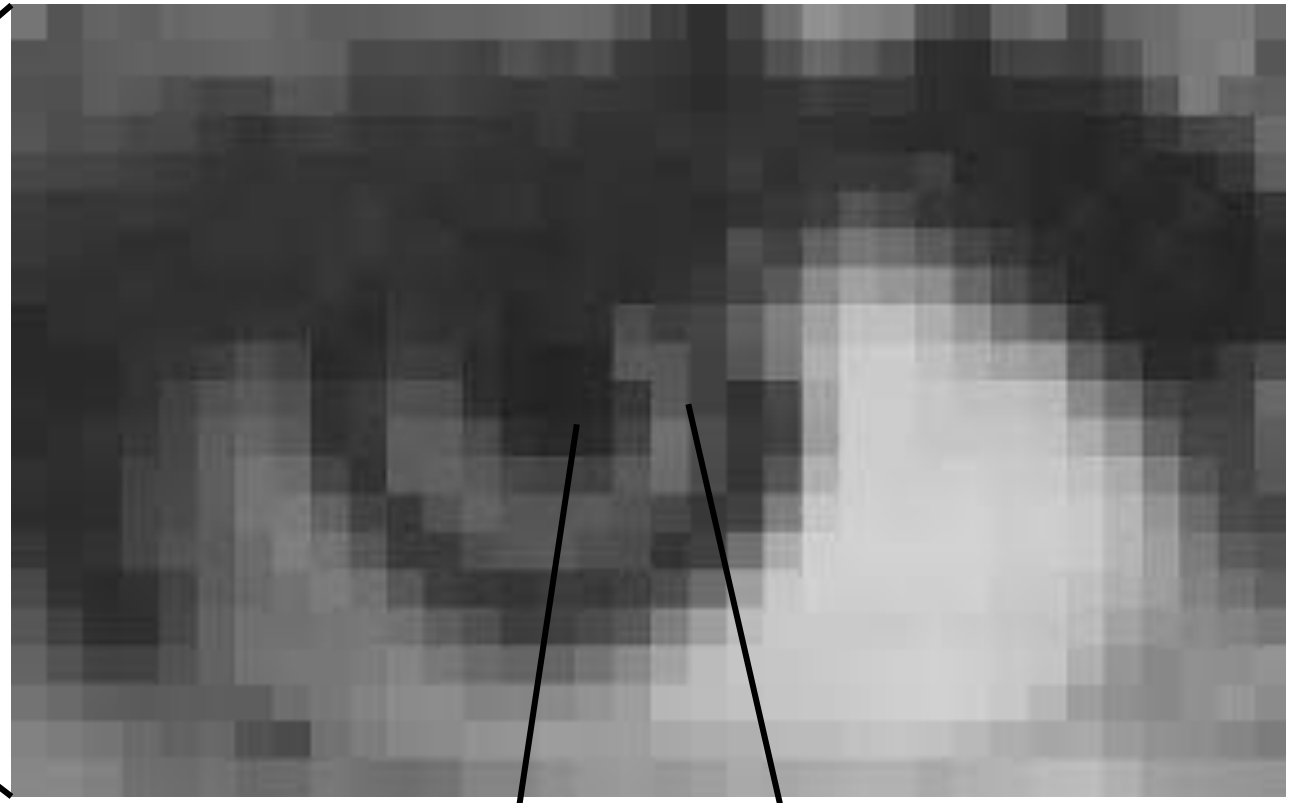
2-D array

2-D array in life

2-D array in life

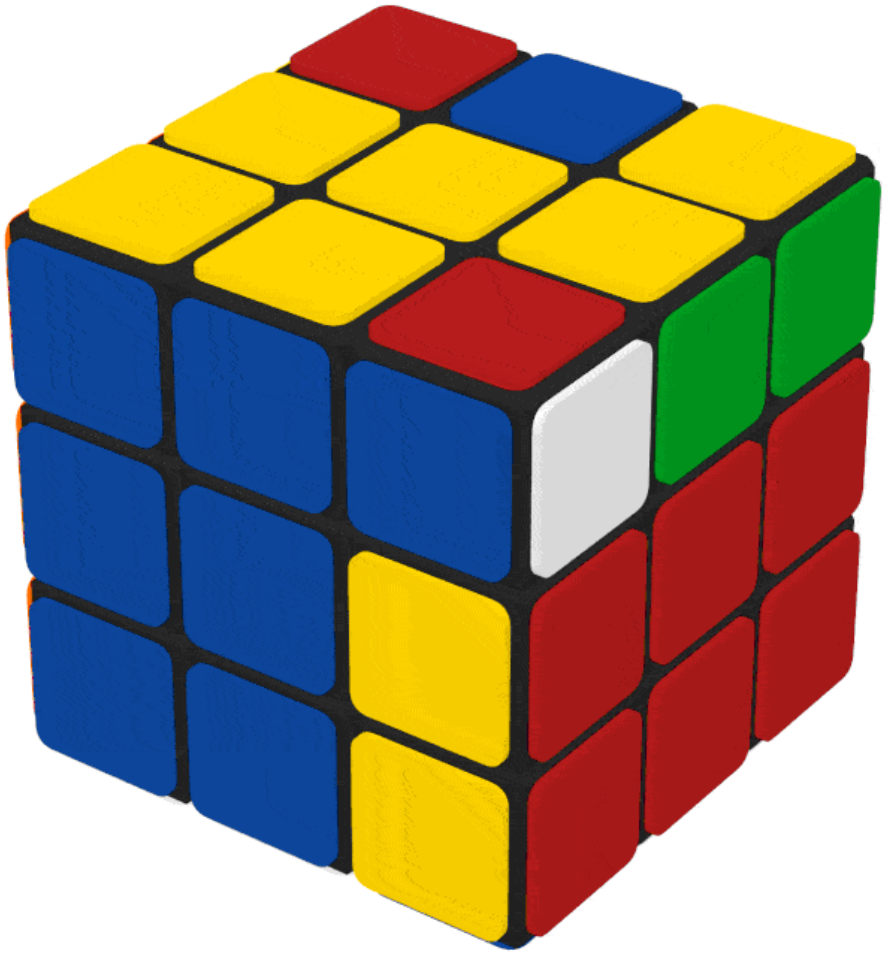
2-D array in life

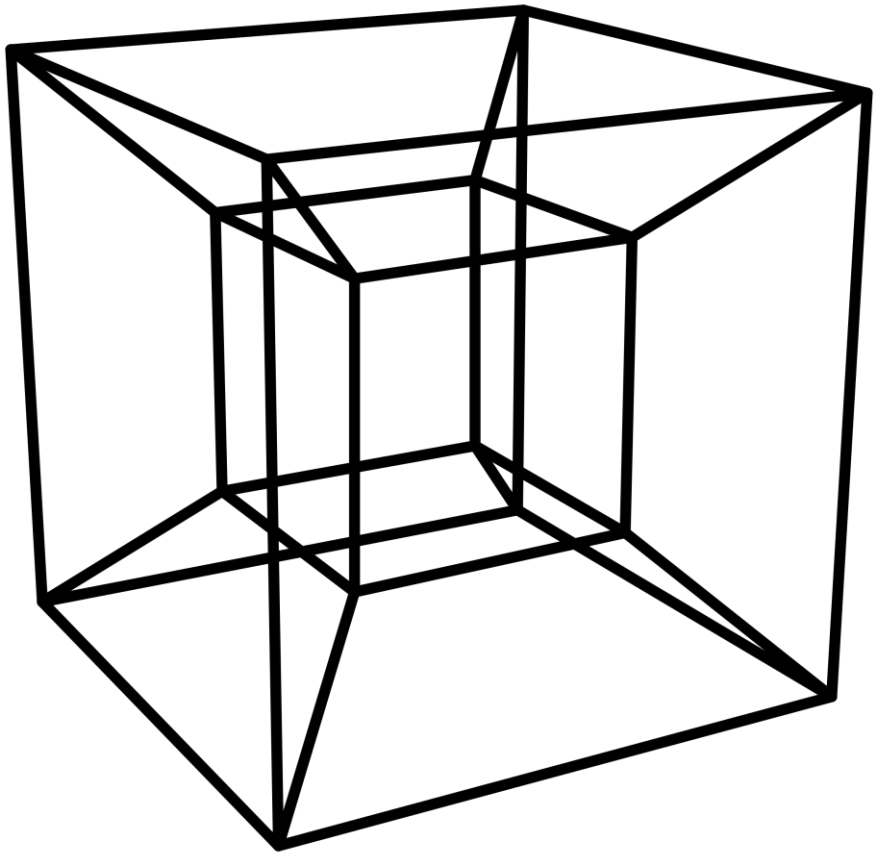
Matrix



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

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

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;

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

Most decisions can be expressed as a linear equation!

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

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
```

Case study: subtract 2 matrices

Case: how to subtract 2 matrices?

```c
#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?

```c
#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

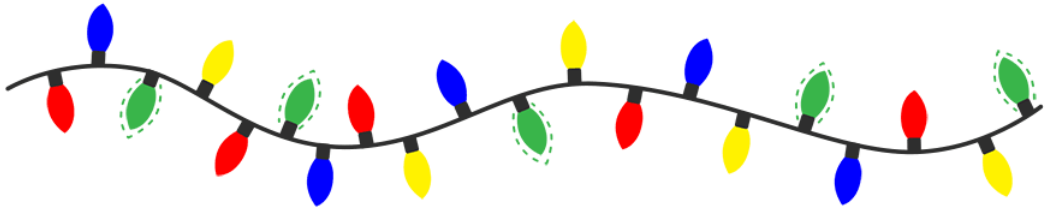
Case: how to cross multiply 2 matrices?

```
#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");
    }
}
```

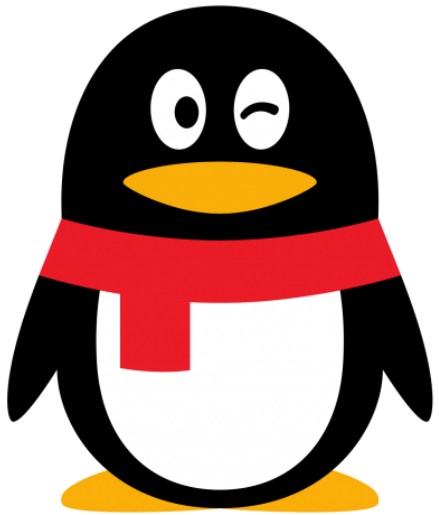
```
Matrix A:
1 2
4 5

Matrix B:
2 2 1
1 3 2

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

Content

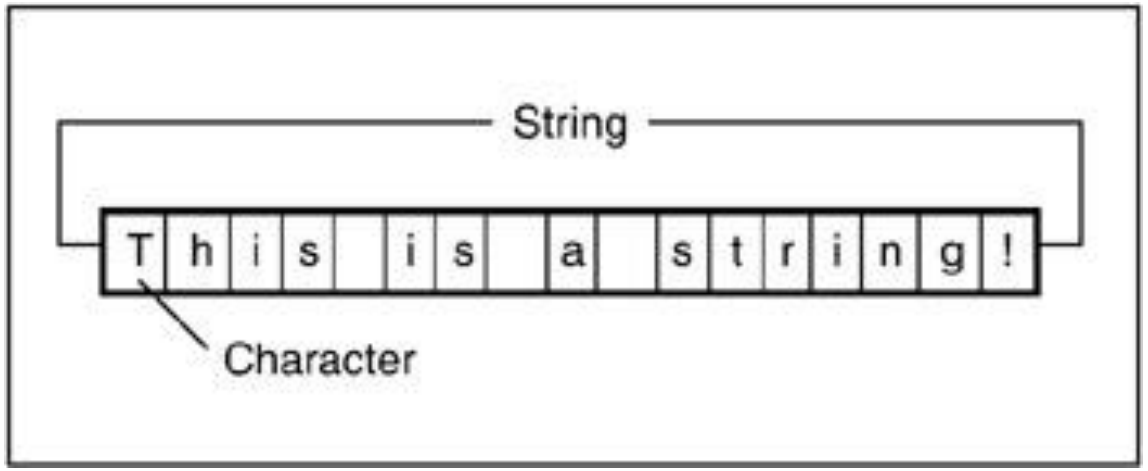
String in life



String in life



QQ



String

String is an array of characters.

```
char name[size] = {'','',…,''};

char name[size] = {"…"};

char name[] = {"…"};

char name[] = "…";
```

String

```
char c[10] = {'I', ' ', 'a', 'm', ' ', 'h', 'a', 'p' , 'p', 'y'}; // length is 10
char c[10] = {"I am happy"};
char c[] = {"I am happy"};
char c[] = "I am happy"; // preferred
```

I		a	m		h	a	p	p	y
c[0]	c[1]	c[2]	c[3]	c[4]	c[5]	c[6]	c[7]	c[8]	c[9]

1D and 2D String

1D char array holds the characters!

char c[10] = “I am happy”;

Machine thinks it as a single “word”!

I		a	m		h	a	p	p	y

2D char array holds the words!

char c[3][10] = {“I”, “am”, “happy”};

Machine thinks it as a group of word!

I									
a	m								
h	a	p	p	y					

String

```
char c[10] = {‘S’, ‘ U’, ‘S’, ‘T’, ‘e ’, ‘c’, ‘h’}; // length is 10
char c[10] = {“SUSTech”};
char c[] = {“SUSTech”};
char c[] = “SUSTech”; // preferred
```

S	U	S	T	e	c	h	\0	\0	\0
c[0]	c[1]	c[2]	c[3]	c[4]	c[5]	c[6]	c[7]	c[8]	c[9]

String operations

C supports a wide range of functions that manipulate strings.

Operators	Description	Example s1=A, S2 = B;
strcpy(s1, s2)	Copy s2 into s1	s1 = B
strcat(s1, s2)	Concatenate s1 and s2	S1 = AB
strlen(s1)	Return length of s2	Length = 1
strcmp(s1, s2)	Compare s1 and s2	A<B, return -1
strlwr(s1)	Convert s1 to lower case	A to a
strupr(s1)	Convert s1 to upper case	A to A

strcpy(s1, s2)

```
char str1[12] = "Hello";
char str2[12] = "World";
char str3[12];

strcpy(str3, str1);
printf("str3 = %s\n", str3); //Hello

strcpy(str3, str2);
printf("str3 = %s\n", str3); //World
```

strcat(s1, s2)

```
char str1[12] = "Hello";
char str2[12] = "World";
char str3[12] = "123";

strcat(str1, str2);
printf("str1 = %s\n", str1); //HelloWorld

strcat(str3, str2);
printf("str3 = %s\n", str3); //123World
```

strlen(s1)

```
char str1[12] = "Hello";
char str2[] = "World";
char str3[12];

printf("str1 = %s\n", strlen(str1)); //5

printf(“str2 = %s\n”, strlen(str2)); //5

printf("str3 = %s\n", strlen(str3)); //0
```

sizeof(s1)

```
char str1[12] = "Hello";
char str2[] = "World";
char str3[12];

printf("str1 = %s\n", sizeof(str1)); //12

printf(“str2 = %s\n”, sizeof(str2)); //6, end with ‘\0’

printf("str3 = %s\n", sizeof(str3)); //12
```

strcmp(s1, s2)

```
char str1[] = "ABCD";
char str2[] = "BCD";
char str3[] = "ABCE";
char str4[] = "1234";

printf("cmp = %d\n", strcmp(str1, str2)); //-1

printf("cmp = %d\n", strcmp(str1, str3)); //-1

printf("cmp = %d\n", strcmp(str1, str1)); //0
```

```
str1 > str2 →  1
str1 < str2 → -1
str1 = str2 →  0
```

strlwr(s1)

```
char str1[] = "ABCD";
char str2[] = "abcd";
char str3[] = “012abcDE";

printf("strlwr = %d\n", strlwr(str1)); //abcd

printf("strlwr = %d\n", strlwr(str2)); //abcd

printf("strlwr = %d\n", strlwr(str3)); //012abcde
```

strupr(s1)

```
char str1[] = "ABCD";
char str2[] = "abcd";
char str3[] = “012abcDE";

printf("strupr = %d\n", strupr(str1)); //ABCD

printf("strupr = %d\n", strupr(str2)); //ABCD

printf("strupr = %d\n", strupr(str3)); //012ABCDE
```

Case study: dictionary

Case: can we create a sentence?

```c
#include <stdio.h>

main()
{
    char greeting[10] = "Hello";
    char greetings[3][10] = { "Hello", "my", "friend" };

    printf("Greeting message: %s\n", greeting);
    printf("Greeting message: %s\n", greetings[1]);
}
```

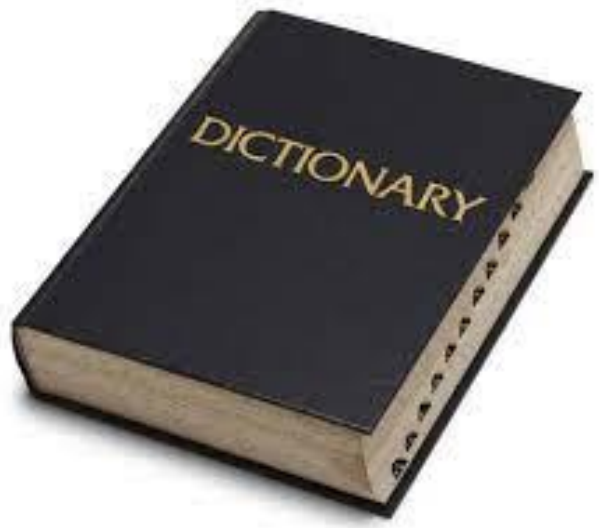
Microsoft Visual Studio Debug Console

```
Greeting message: Hello
Greeting message: my
```

Case study: dictionary

Case: can we create a simple dictionary?

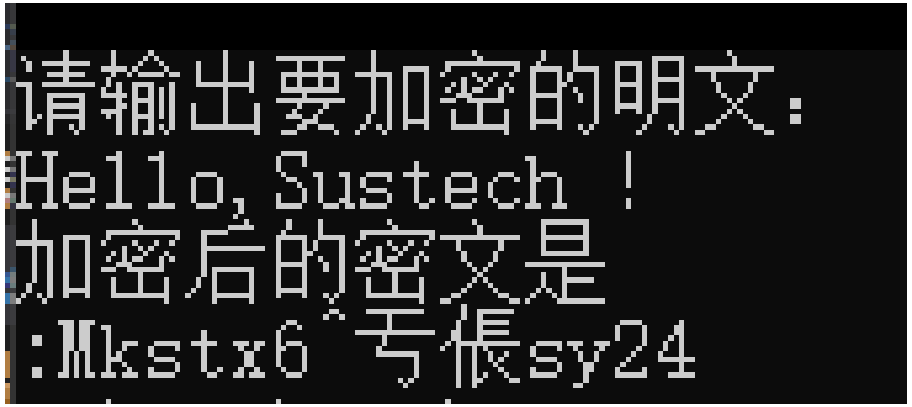
```
#include<stdio.h>
#include<string.h>
main() {
        char EngWords[][8] = { "apple","orange","banana" };
        char ChineseWords[][8] = { "苹果","橘子","香蕉"};
        char text[128];
        while (gets(text))
        {
        for (int i = 0;i < 3;i++) {
        if (strcmp(text, EngWords[i])==0) {
                printf("%s 中文为: %s\n", text,ChineseWords[i]);
                break;}
        if (strcmp(text, ChineseWords[i])==0) {
                printf("%s 英文为 %s\n", text, EngWords[i]);
                break;}
        }
        if (strcmp(text, "exit")==0)break;
        }
}
```



Case study: encryption

Case: can we encrypt a message?

```
#include<stdio.h>
#include<string.h>

main() {
        char text[128] = { '\0' };
        char cryptograph[128] = { '\0'};
        printf("请输出要加密的明文: \n");
        gets(text);
        int count = strlen(text);
        for (int i = 0; i < count; i++)
        {
                cryptograph[i] = text[i] + i + 5;
        }
        cryptograph[count] = '\0';
        printf("加密后的密文是\n:%s", cryptograph);
}
```



Summary

- We can use **array** to hold data for group processing
- Array has the **fixed size** and can only be used to hold data with **same type**
- **Different types of array** can be created, e.g. int array, float array, char array (string)
- **Different dimensional array** can be created, from 1D array to ND array
- Array enables the processing of **vectors, matrices, strings**, etc.
- Time to write you C program to process a group of data.

Assignment

1. Enter 5 float numbers, store them in an array and print their reciprocals in the reverse order

a) Use “scanf” to enter the float numbers
b) Test input : 0.5 , 0.9 , 1.3 , 4.7 , 9.3

2. Enter three strings, convert “china” from lowercase to uppercase, combine them into one string and print how many words are in it

a) Use “scanf” to enter the strings
b) Test input : “I” , “love” , “china”

Assignment

3. Create a 2D float array to store matrix A $\begin{matrix} 14 & 45 \\ 34 & 43 \\ 54 & 67 \\ 67 & 26 \end{matrix}$, calculate $A \cdot A^T$ and downsize $A \cdot A^T$ to a 2x2 matrix and print it

a) $A \cdot A^T$ is a 4*4 matrix

b) A^T is the transpose of A

c) Downsize means take the average of adjacent elements as a new element

$$\begin{matrix} 14 & 45 & 87 & 89 \\ 34 & 43 & 56 & 43 \\ 54 & 67 & 76 & 9 \\ 67 & 26 & 78 & 23 \end{matrix}$$

$$34.00 = \frac{14 + 45 + 34 + 43}{4}$$

$$\begin{matrix} \boxed{34.00} & 68.75 \\ 53.50 & \boxed{46.50} \end{matrix}$$

$$46.50 = \frac{76 + 9 + 78 + 23}{4}$$

Assignment

4. Enter a date in the format such as: 2010 10 24 and print the number of day of the year.

a) For example, 2010 10 24 is 297th day of 2010
b) Test input: 2022 9 30

5. Use insertion sort (mentioned in the 1D array operation) to sort 15 integers and print them in the ascending order (from min to max)

a) The 15 integers are 3,44,38,5,47,15,36,26,27,2,46,4,19,50,48

Assignment

6. (**bonus**) There are 100 lights (numbered 1 ~ n), the 1st person turns on all lights; the 2nd person presses all switches numbered in multiples of 2; the 3rd person presses all switches numbered in multiples of 3; the 4th person presses all switches numbered in multiples of 4 …… There are 100 people in total. Which lights are on in the end?

a) When a person presses the switch of a light, the light will be turned on if it was off or turned off if it was on
b) Print the index of lights which are on in the end
c) Multiples of 3 means 3, 6, 9, 12, 15, 18 …; multiples of 5 means 5, 10, 15, 20, …