Introduction to C Programming

Lecture 12: review III

Wenjin Wang
wangwj3@sustech.edu.cn

12-9-2022

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	I/O	2022.10.28

Nr.	Lecture	Date
9	Head files	2022.11.4
10	Review of lectures I	2022.11.25
11	Review of lectures II	2022.12.2
12	Review of lectures III	2022.12.9
13	AI in C programming	2022.12.16
14	AI in C programming	2022.12.23
15	Summary	2023.12.30

Course syllabus

Review of lectures I

Review of lectures II

Review of lectures III

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	I/O	2022.10.28
9	Head files	2022.11.4

Objective of this lecture

Review the learned lectures 6 – 9:

Pointer, Self-defined types, I/O, head files

Content

Content

1. Pointer
2. Self-defined types
3. I/O
4. Head files

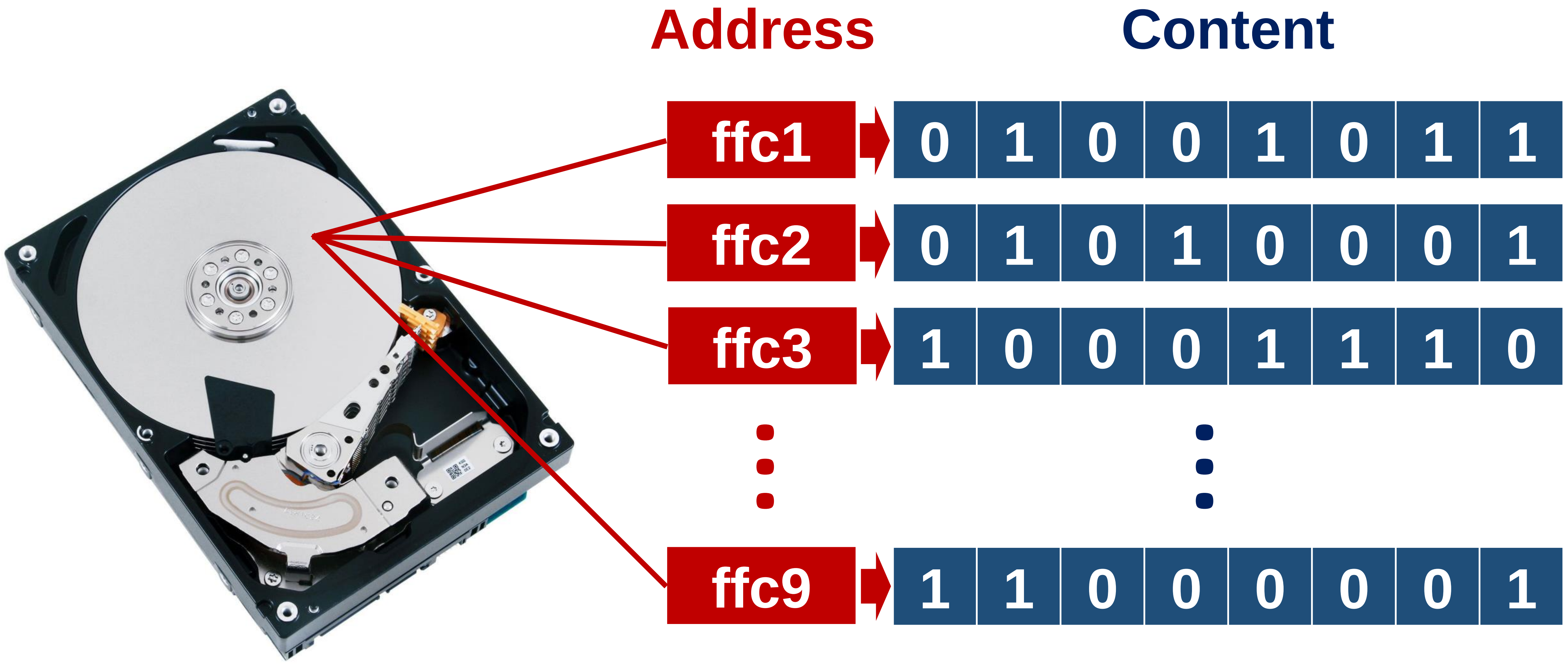
Memory address

The memory address is the location of where the variable is stored on a PC.

When a variable is created in C, a memory address is assigned to the variable.

When we assign a value to the variable, it is stored at this memory address.

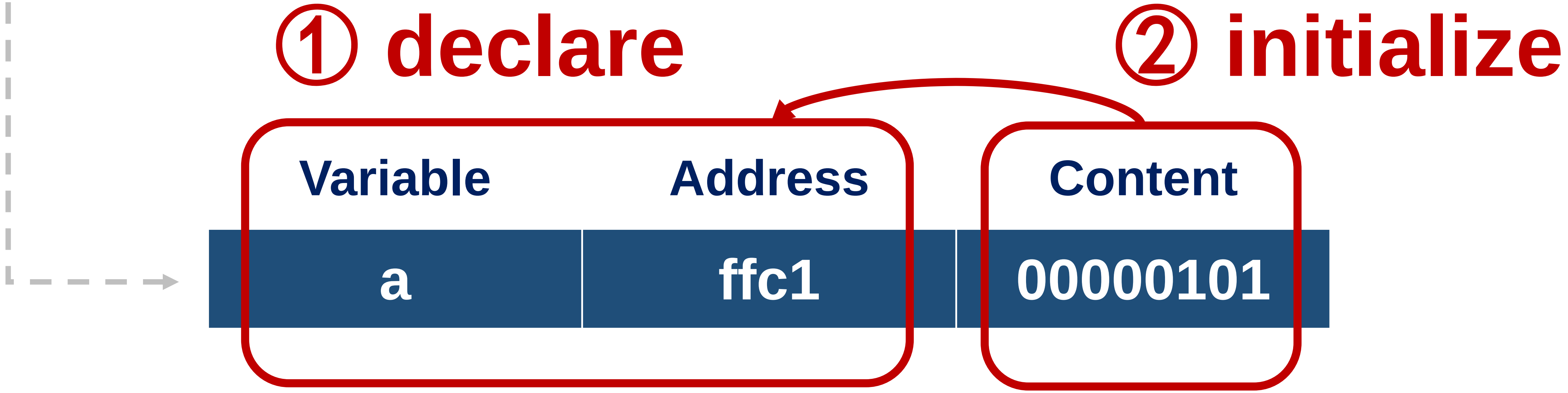
Memory address



Address	Content							
ffc1	0	1	0	0	1	0	1	1
ffc2	0	1	0	1	0	0	0	1
ffc3	1	0	0	0	1	1	1	0
⋮	⋮							
ffc9	1	1	0	0	0	0	0	1

Memory address

```
int a = 5;  { int a;//declare
            { a = 5;//initialize
```



Memory address

```
int a = 5;
int b = 2;
int c = 1;
```

Variable	Address	Content
a	ffc1	00000101
b	ffc2	00000010
c	ffc3	00000001

What happens in the memory allocation?

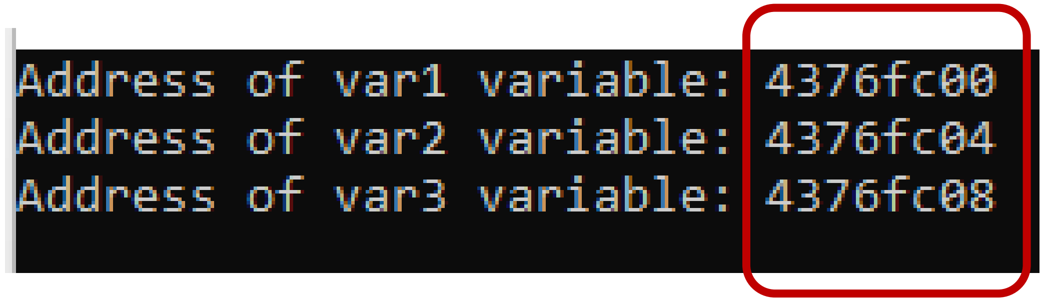
How to check variable address

Use **& (reference operator)** to check the variable address

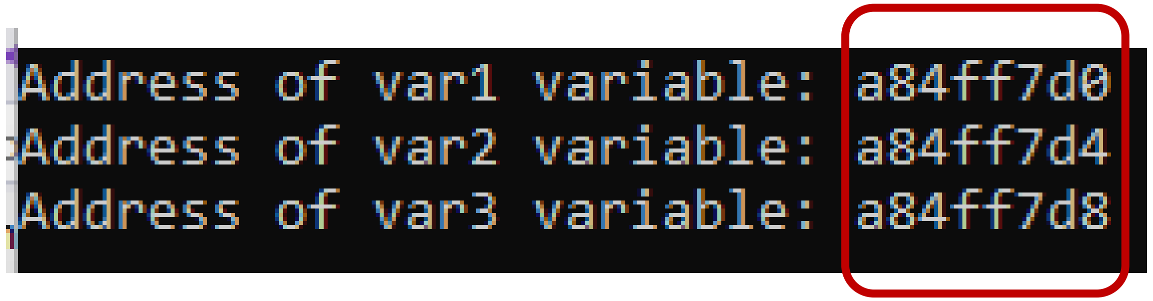
```
#include <stdio.h>

main ()
{
   int var1;
   float var2;
   char var3;
   printf("Address of var1 variable: %x\n", &var1);
   printf("Address of var2 variable: %x\n", &var2);
   printf("Address of var3 variable: %x\n", &var3);
}
```

How to check variable address

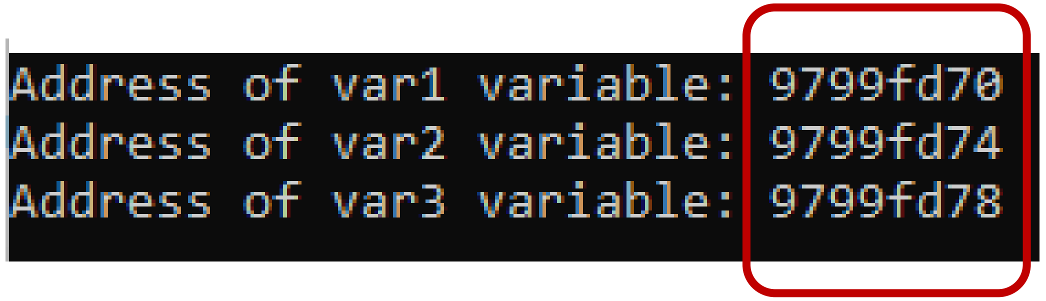
Run multiple times, every time the address is different, but it has orders!



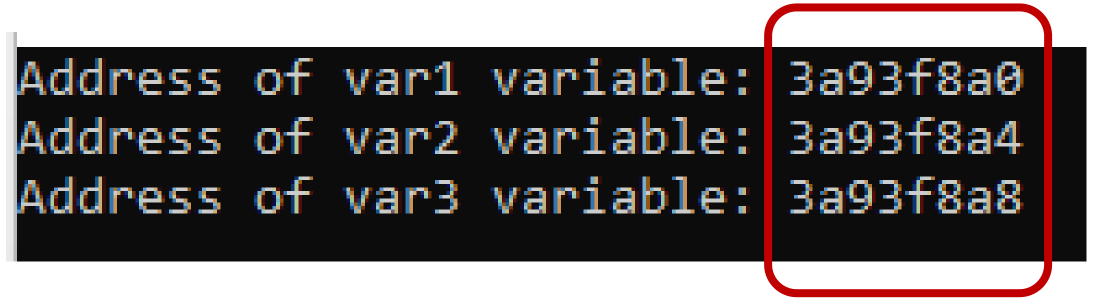
```
Address of var1 variable: 4376fc00
Address of var2 variable: 4376fc04
Address of var3 variable: 4376fc08
```



```
Address of var1 variable: a84ff7d0
Address of var2 variable: a84ff7d4
Address of var3 variable: a84ff7d8
```



```
Address of var1 variable: 9799fd70
Address of var2 variable: 9799fd74
Address of var3 variable: 9799fd78
```



```
Address of var1 variable: 3a93f8a0
Address of var2 variable: 3a93f8a4
Address of var3 variable: 3a93f8a8
```

What is Hexadecimal?

Decimal number system

↓

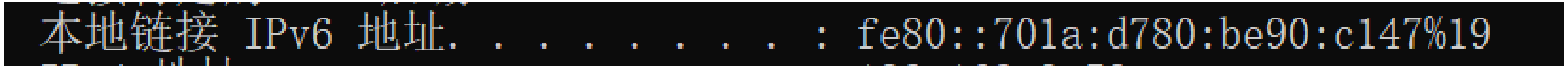
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

0 1 2 3 4 5 6 7 8 9 A B C D E F 10

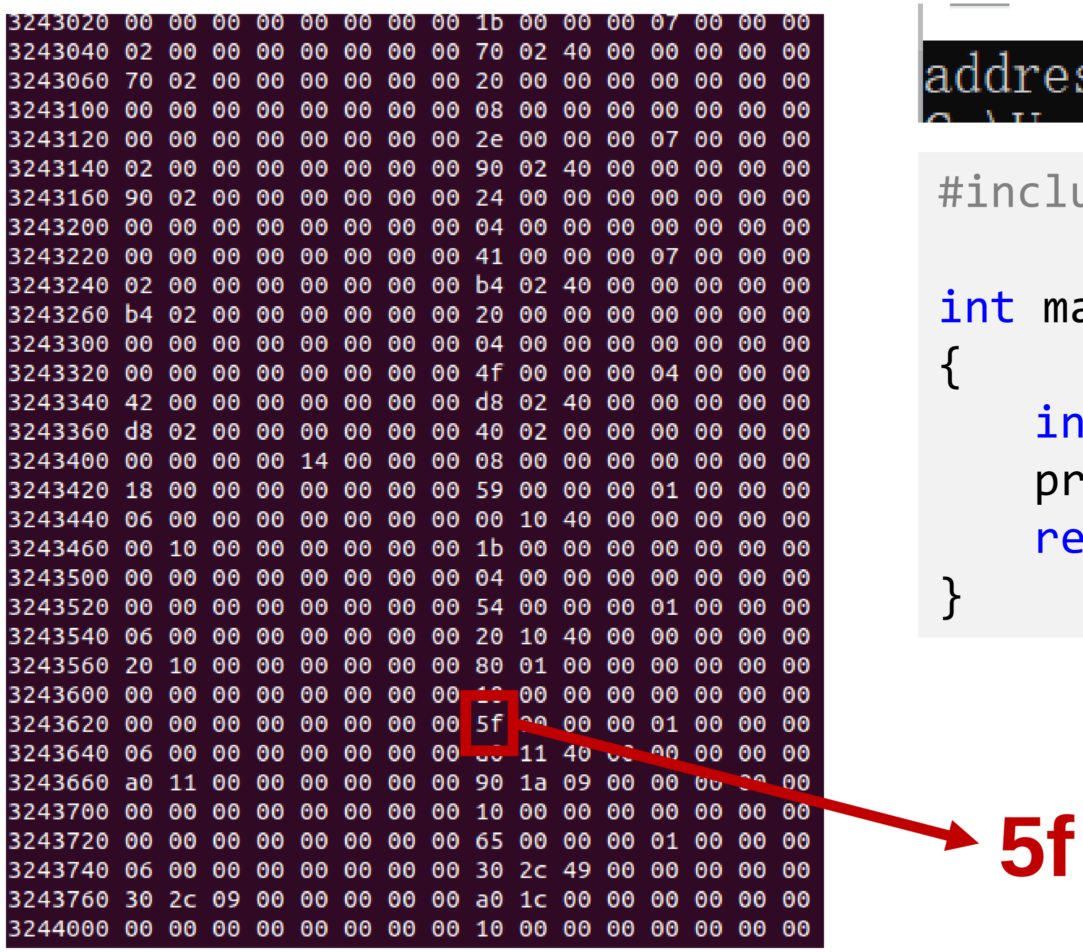
↑

Hexadecimal number system

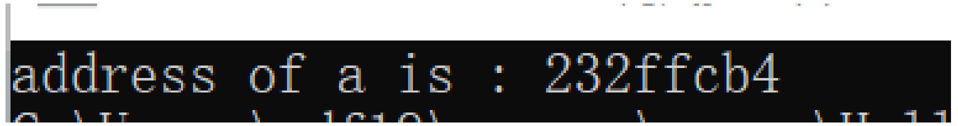
Hexadecimal is everywhere



```
本地链接 IPv6 地址. . . . . . . . : fe80::701a:d780:be90:c147%19
```



```
3243020 00 00 00 00 00 00 00 00 1b 00 00 00 07 00 00 00
3243040 02 00 00 00 00 00 00 00 70 02 40 00 00 00 00 00
3243060 70 02 00 00 00 00 00 00 20 00 00 00 00 00 00 00
3243100 00 00 00 00 00 00 00 00 08 00 00 00 00 00 00 00
3243120 00 00 00 00 00 00 00 00 2e 00 00 00 07 00 00 00
3243140 02 00 00 00 00 00 00 00 90 02 40 00 00 00 00 00
3243160 90 02 00 00 00 00 00 00 24 00 00 00 00 00 00 00
3243200 00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00
3243220 00 00 00 00 00 00 00 00 41 00 00 00 07 00 00 00
3243240 02 00 00 00 00 00 00 00 b4 02 40 00 00 00 00 00
3243260 b4 02 00 00 00 00 00 00 20 00 00 00 00 00 00 00
3243300 00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00
3243320 00 00 00 00 00 00 00 00 4f 00 00 00 04 00 00 00
3243340 42 00 00 00 00 00 00 00 d8 02 40 00 00 00 00 00
3243360 d8 02 00 00 00 00 00 00 40 02 00 00 00 00 00 00
3243400 00 00 00 00 14 00 00 00 08 00 00 00 00 00 00 00
3243420 18 00 00 00 00 00 00 00 59 00 00 00 01 00 00 00
3243440 06 00 00 00 00 00 00 00 00 10 40 00 00 00 00 00
3243460 00 10 00 00 00 00 00 00 1b 00 00 00 00 00 00 00
3243500 00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00
3243520 00 00 00 00 00 00 00 00 54 00 00 00 01 00 00 00
3243540 06 00 00 00 00 00 00 00 20 10 40 00 00 00 00 00
3243560 20 10 00 00 00 00 00 00 80 01 00 00 00 00 00 00
3243600 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00 00
3243620 00 00 00 00 00 00 00 00 5f 00 00 00 01 00 00 00
3243640 06 00 00 00 00 00 00 00 a0 11 40 00 00 00 00 00
3243660 a0 11 00 00 00 00 00 00 90 1a 09 00 00 00 00 00
3243700 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00 00
3243720 00 00 00 00 00 00 00 00 65 00 00 00 01 00 00 00
3243740 06 00 00 00 00 00 00 00 30 2c 49 00 00 00 00 00
3243760 30 2c 09 00 00 00 00 00 a0 1c 00 00 00 00 00 00
3244000 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00 00
```



```
address of a is : 232ffcb4
```

```c
#include<stdio.h>

int main()
{
    int a = 5;
    printf("address of a is : %x",&a);
    return 0;
}
```

What is pointer?

Variable $\longrightarrow$ Address

指针：存储地址的变量

What is pointer?

Pointer is a variable that stores the address of another variable.

```
type var1;
type *var2 = &var1;
```

```
int a;
float f;
char c;
```

$\Big\}$ Stores value

```
int *a;
float *f;
char *c;
```

$\Big\}$ Stores address

What is pointer?

```
int a;
```

- a has type of int
- a stores value

```
int *b;
```

- b has type of int*
- b stores address

What is pointer?

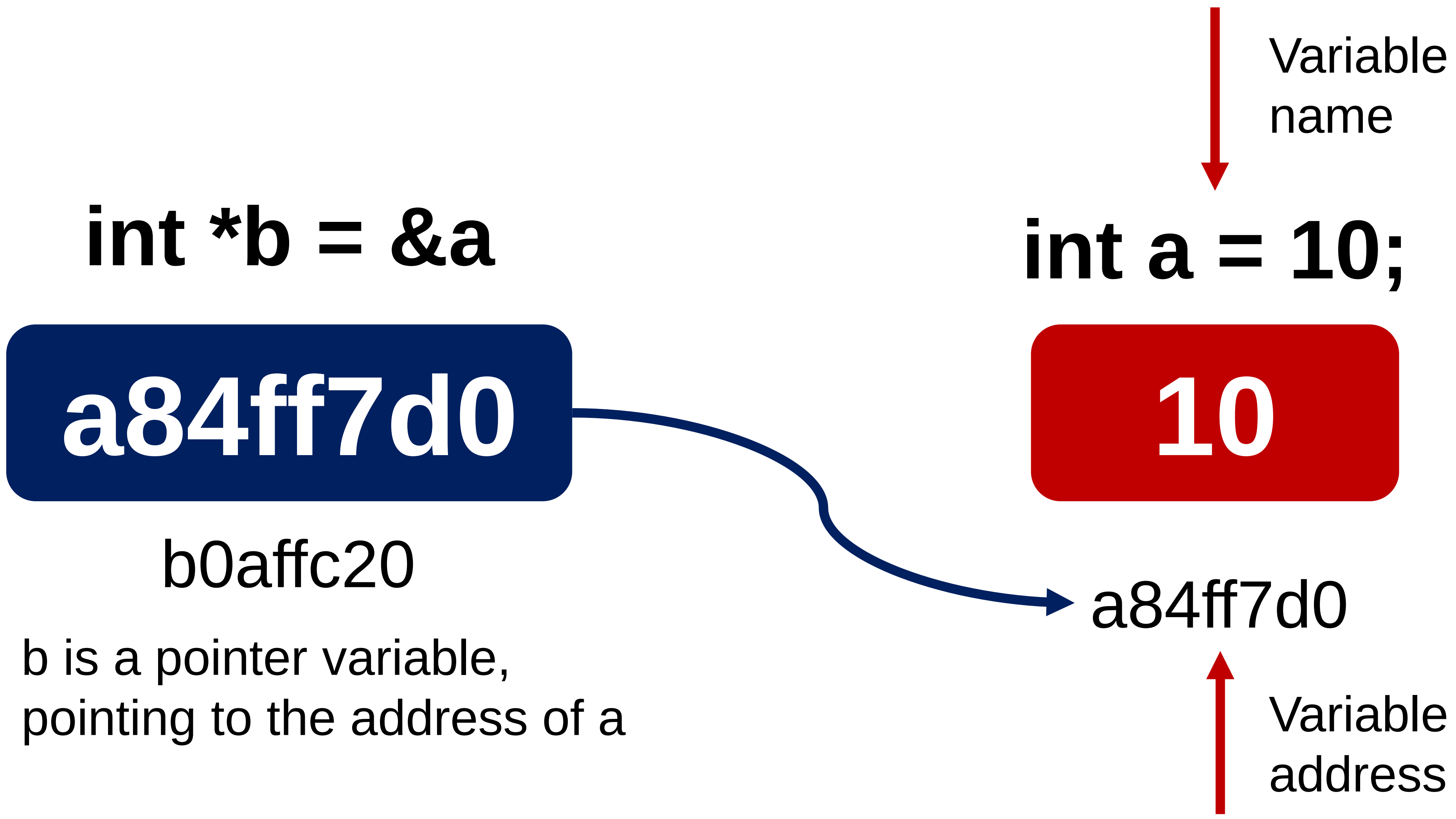
a stores an integer value

```
int a = 10;

int *b = &a;
```

b stores the address of an integer variable

Get the address

What is pointer?



```
int *b = &a
```

```
int a = 10;
```

b is a pointer variable, pointing to the address of a

What is pointer?

```
int a = 5;
int *b = &a;
```

Variable	Address	Content
a	ffc1	00001010
b	ffc2	ffc1

- a stores the value of 10
- b stores the address of a

How to interpret pointer?

int *b

b has data type int*

```
printf("%x", b);//address
```

int *b

*b has data type int

```
printf("%d", *b);//value
```

How to interpret pointer?

```
int a = 5;
int *b = &a;
```

Use **b** to check the address of a

Use ***b** to check the value of a

How to define pointer?

Pointer stores address, not value!

```
int a = 5;
int *b = &a;
```

✓

```
int a = 5;
int *b = 10;
```

✗

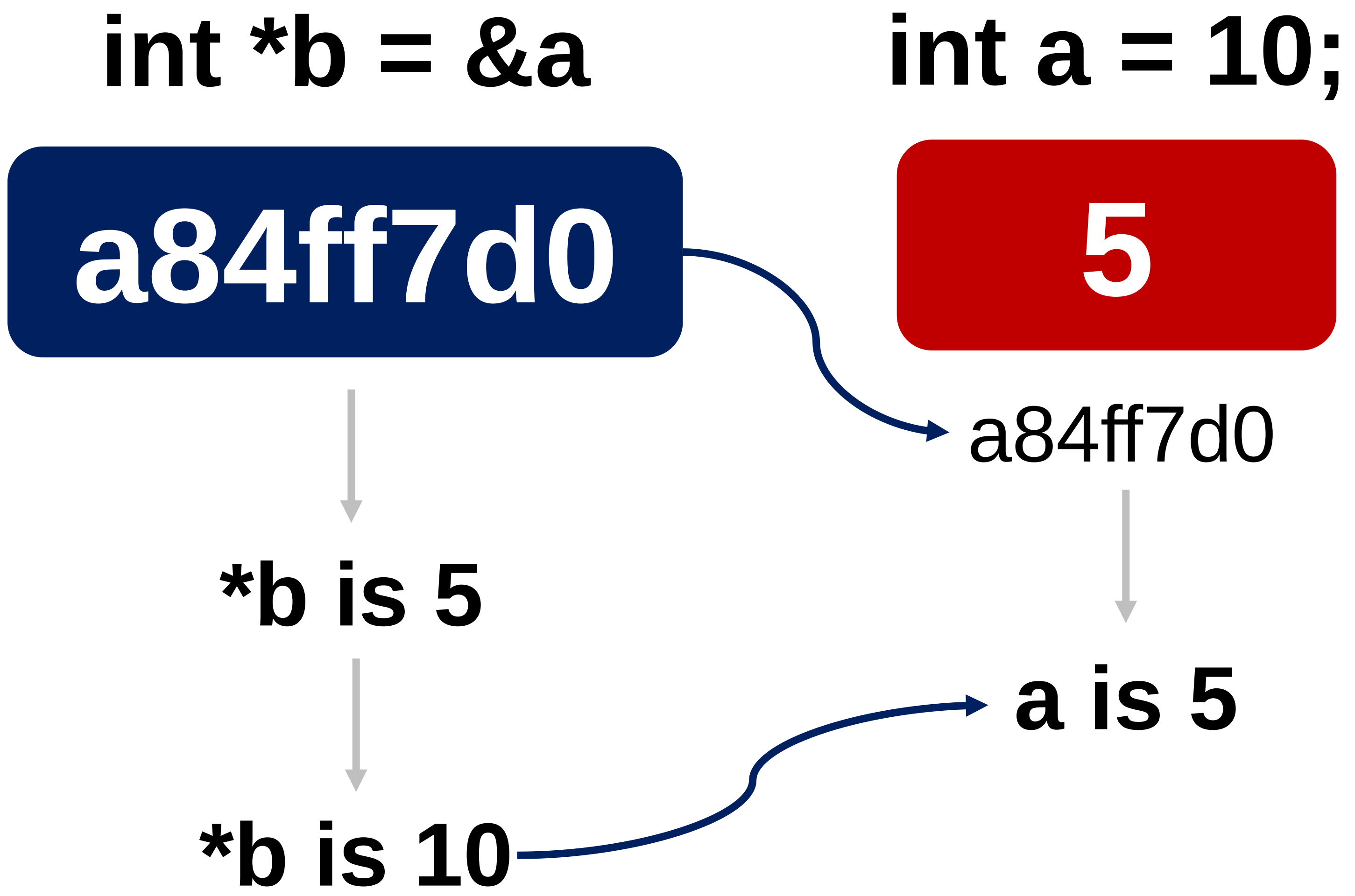
```
int a = 5;
int *b = &a;
*b = 10;
```

✓

How to define pointer?

```
int a = 5;
int *b = &a;

*b = 10;
```

What is a?



How to use pointer?

1. Use pointer for functions to pass values

2. Use pointer for array operations (elements in an array has continuous address)

Use pointer for functions

Values cannot be swapped

```
void swap(int v1, int v2)
{
   printf(“Before: v1=%d, v2=%d\n", v1, v2);
   int temp;
   temp = v1;
   v1 = v2;
   v2 = temp;
   printf(“After: v1=%d, v2=%d\n", v1, v2);
}

main()
{
    int a = 10, b = 5;
    printf(“Before: a=%d, b=%d\n", a, b);
    swap(a, b);
    printf(“After: a=%d, b=%d\n", a, b);
}
```

Variable	Address	Content
a	ffc1	10
b	ffc2	5

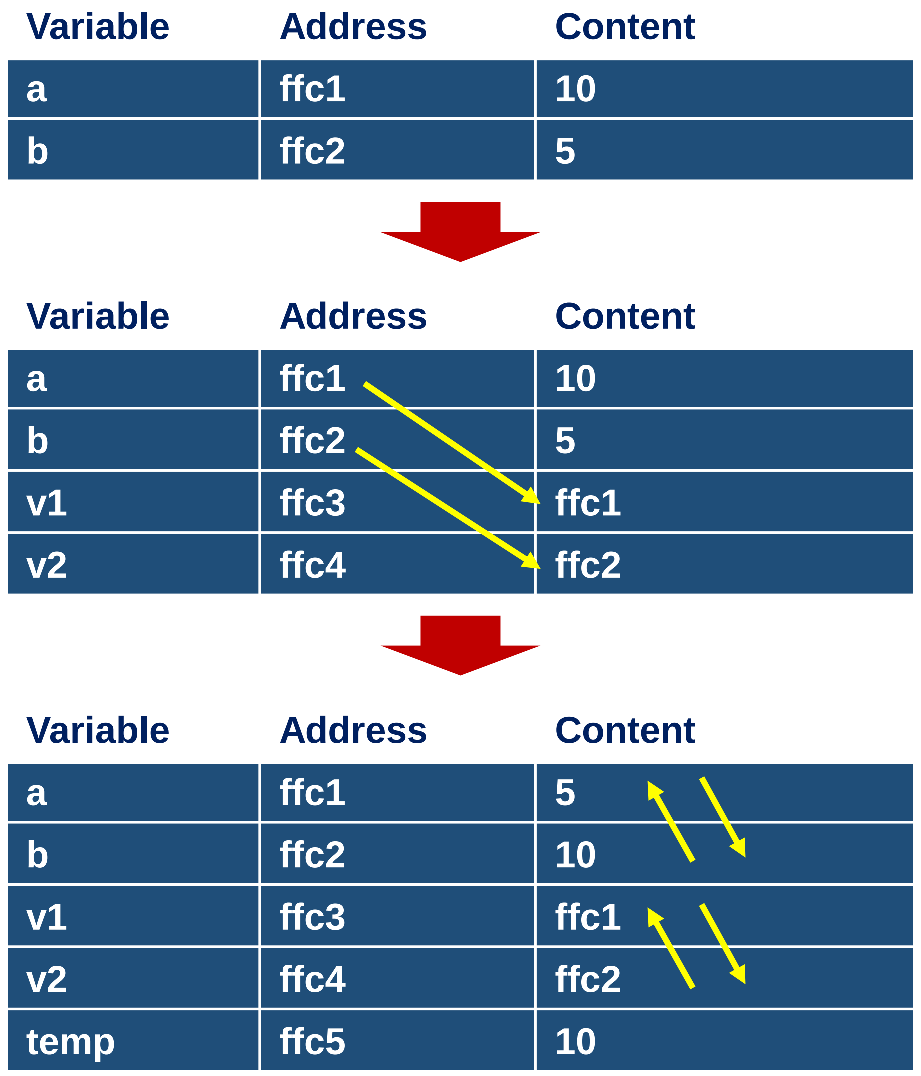
Variable	Address	Content
a	ffc1	10
b	ffc2	5
v1	ffc3	10
v2	ffc4	5

Variable	Address	Content
a	ffc1	10
b	ffc2	5
v1	ffc3	5
v2	ffc4	10
temp	ffc5	10

Use pointer for functions

Values can be swapped

```
void swap(int *v1, int *v2)
{
   int temp;
   temp = *v1;
   *v1 = *v2;
   *v2 = temp;
}

main()
{
    int a = 10, b = 5;
    printf("Before: a=%d, b=%d\n", a, b);
    swap(&a, &b);
    printf("After: a=%d, b=%d\n", a, b);
}
```



Variable	Address	Content
a	ffc1	10
b	ffc2	5

Variable	Address	Content
a	ffc1	10
b	ffc2	5
v1	ffc3	ffc1
v2	ffc4	ffc2

Variable	Address	Content
a	ffc1	5
b	ffc2	10
v1	ffc3	ffc1
v2	ffc4	ffc2
temp	ffc5	10

Use pointer for functions

How to output multiple results from a function?

```
int func(int v1, int v2)
{
    int v3 = v1 + v2;
    int v4 = v1 - v2;
    return v3;
}

main()
{
    int a = 10, b = 5;
    int c = func(a, b);
}
```

We did multiple operations but only return one result!

Use pointer for functions

How to output multiple results from a function?

```
void func(int v1, int v2, int* sum, int* sub, int* mul, int* div)
{
    *sum = v1 + v2;
    *sub = v1 - v2;
    *mul = v1 * v2;
    *div = v1 / v2;
}

main()
{
    int a = 10, b = 5, sum, sub, mul, div;
    int sum = func(a, b, &sum, &sub, &mul, &div);
}
```

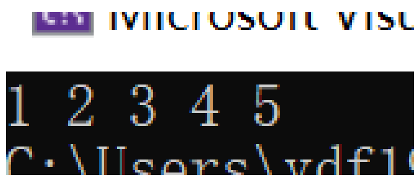
Pass out four results

Use pointer for functions

```
int * myFunction()
{

   ...

}
```

```
int* merge(int a, int b, int c, int d, int e)
{
    int* array = (int*)malloc(sizeof(int) * 5);
    array[0] = a;
    array[1] = b;
    array[2] = c;
    array[3] = d;
    array[4] = e;
    return array;
}

main()
{
    int* array = merge(1, 2, 3, 4, 5);
    for (int i = 0; i < 5; i++)
    printf("%d ", array[i]);
}
```

动态数组



```
Microsoft Visu
1 2 3 4 5
C:\Users\ydf1
```

Pointer points to array

```c
int a = 5;
int *b = &a;
```

$\downarrow$

Give the address of a to b!

```c
int a[10];
int *b = a;
```

$\downarrow$

Give the address of **first element of a** to b!

```c
int *b = &a[0];
```

Pointer points to array

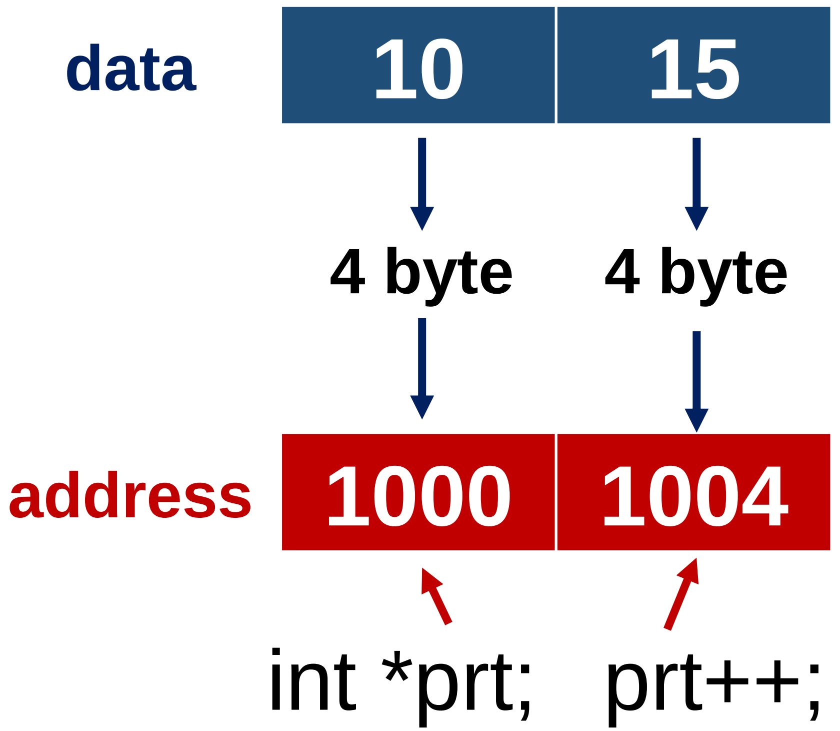
```
int a[3]={1,2,3};

int *b = a;
or
int *b = &a[0];
```

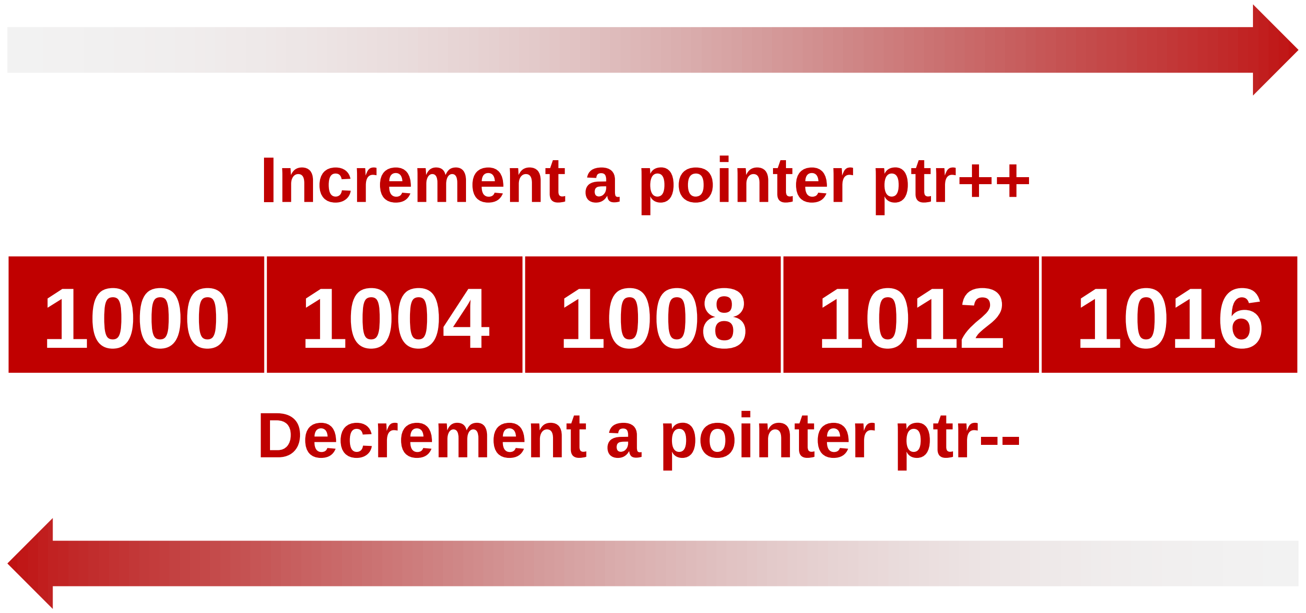
Array	Address	Content
a[0]	17d8f780	1
a[1]	17d8f784	2
a[2]	17d8f788	3
…	…	

Address of the first element is assigned to pointer

Pointer points to array

Four arithmetic operators that can be used on pointers: ++, --, +, -





Pointer points to array

How to access the elements in array?

```
int a[10];
int *b = a;
```

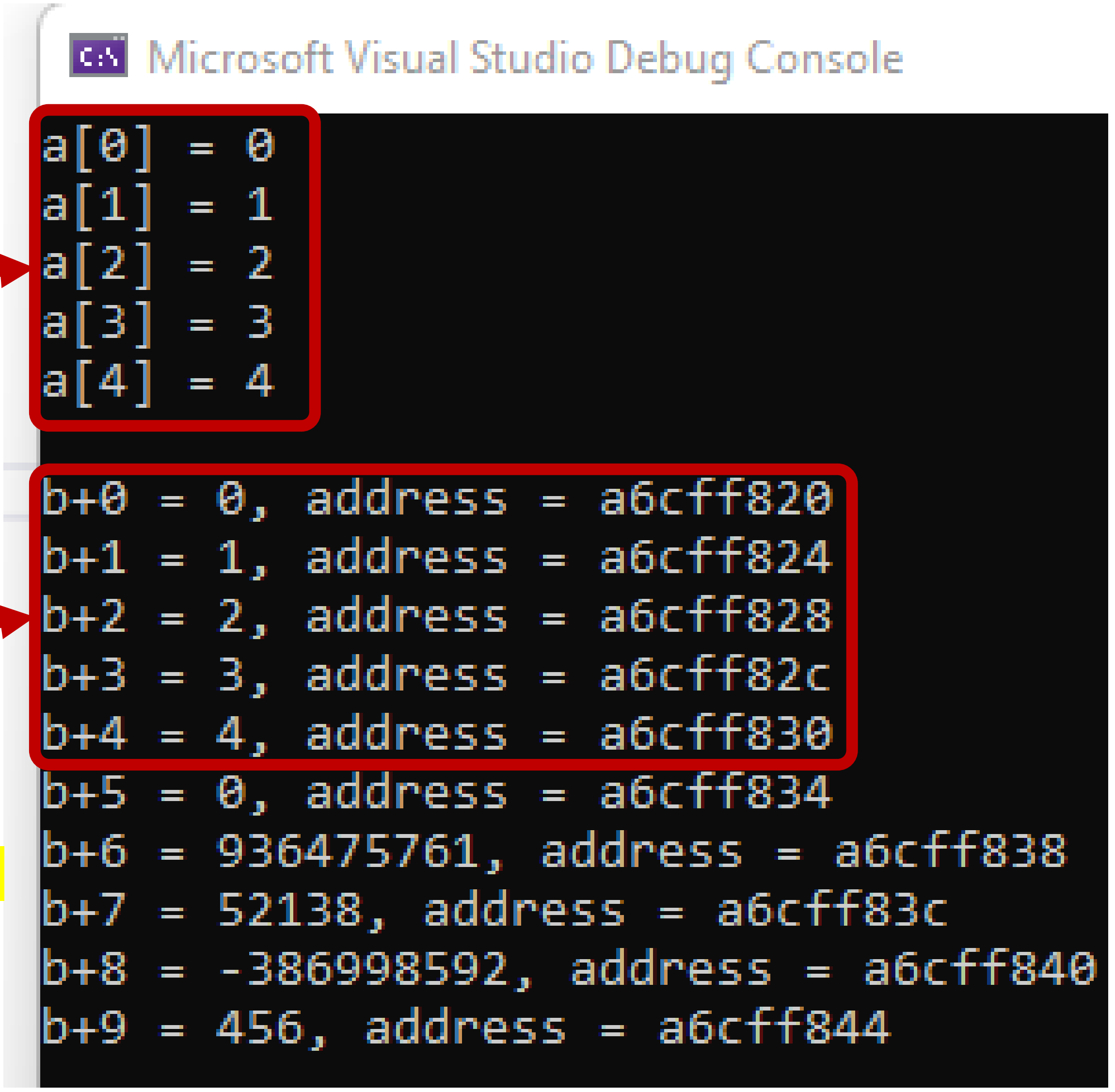
下标法： a[5]

指针法： *(b+5)

Use pointer for array

Use pointer to access array

```
#include<stdio.h>
main()
{
    int a[5] = {0, 1, 2, 3, 4};
    int* b = a;

    for (int i = 0; i < 5; i++)
    {
        printf("a[%d] = %d\n", i, a[i]);
    }

    for (int i = 0; i < 10; i++)
    {
        printf("b+%d = %d, address = %x\n", i, 
*(b+i), b+i);
    }
}
```

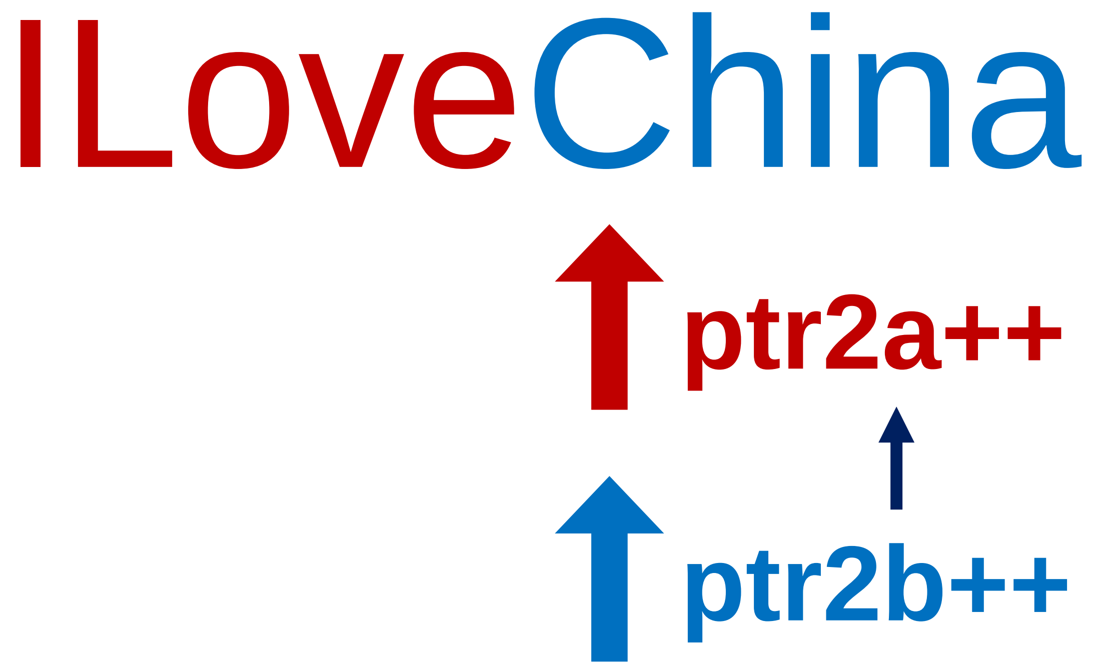

Microsoft Visual Studio Debug Console

```
a[0] = 0
a[1] = 1
a[2] = 2
a[3] = 3
a[4] = 4

b+0 = 0, address = a6cff820
b+1 = 1, address = a6cff824
b+2 = 2, address = a6cff828
b+3 = 3, address = a6cff82c
b+4 = 4, address = a6cff830
b+5 = 0, address = a6cff834
b+6 = 936475761, address = a6cff838
b+7 = 52138, address = a6cff83c
b+8 = -386998592, address = a6cff840
b+9 = 456, address = a6cff844
```

Use pointer for array

How to concatenate 2 strings?

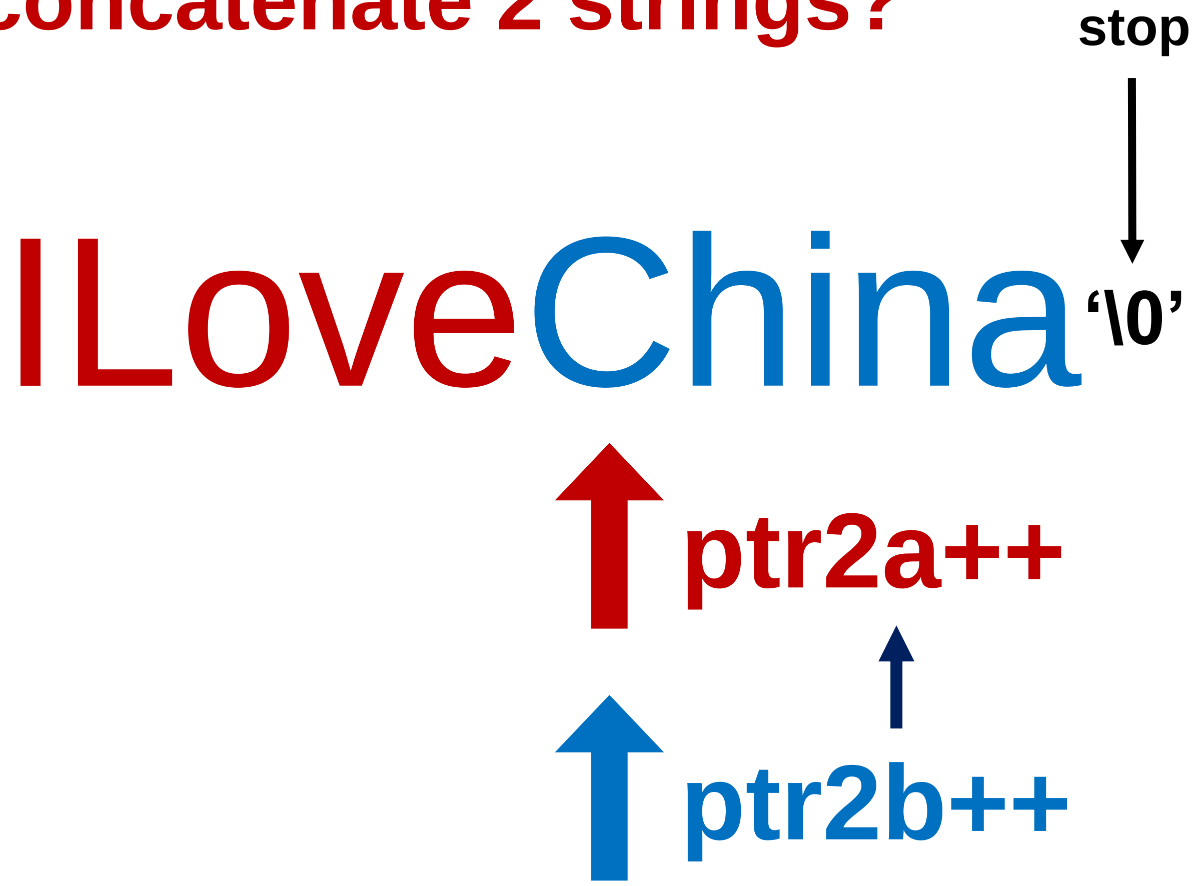
```
#include<stdio.h>

main()
{
    char a[100] = "ILove";
    char b[] = "China";

    char* ptr2a = &a[5]; // last address + 1
    char* ptr2b = &b[0]; // first address

    for (int i = 0; i < sizeof(b); i++)
    {
        *ptr2a = *ptr2b;
        ptr2a++;
        ptr2b++;
    }

    printf("%s\n", a);
}
```



Use pointer for array

```
#include<stdio.h>

main()
{
    char a[100] = "ILove";
    char b[] = "China";

    char* ptr2a = &a[5]; // last address + 1
    char* ptr2b = &b[0]; // first address

    while (*ptr2b != '\0')
    {
        *ptr2a = *ptr2b;
        ptr2a++;
        ptr2b++;
    }

    printf("%s\n", a);
}
```

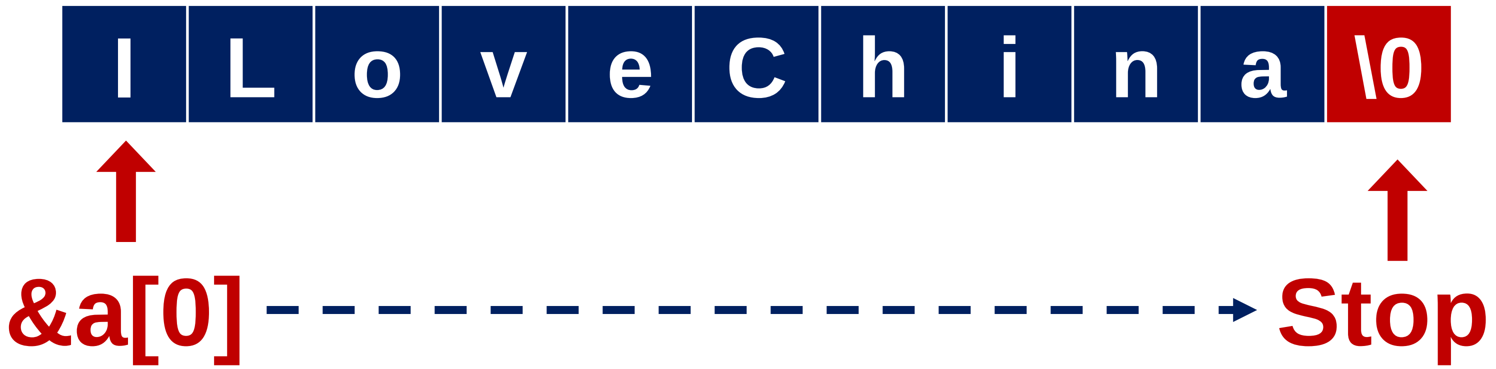
How to concatenate 2 strings?



Use pointer for array

```
#include<stdio.h>

main()
{
    char a[100] = "ILoveChina";
    char* ptr2a = &a[0];

    int length = 0;
    while(*ptr2a != '\0’)
    {
        ptr2a++;
        length++;
    }

    printf("Length of a is %d\n", length);
}
```

What is the length of a string?



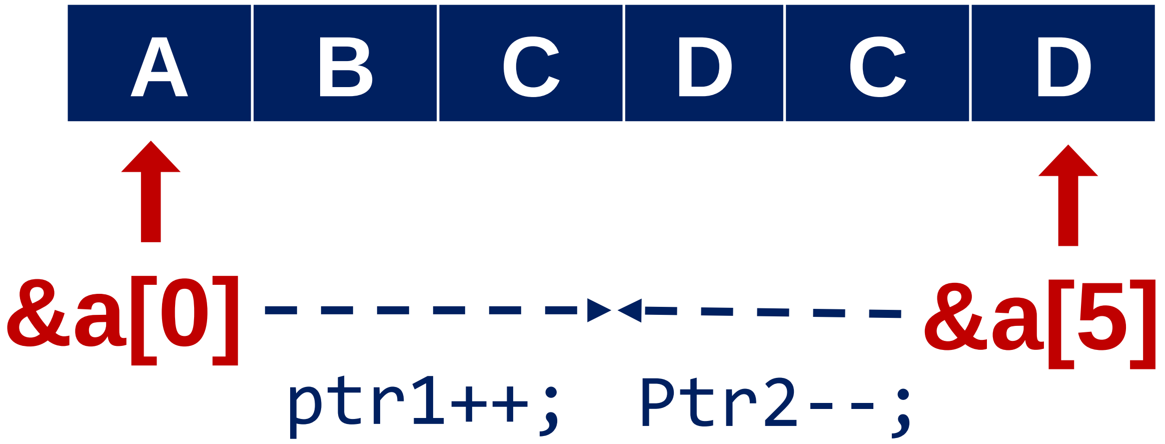
ptr2a++;

Length = 10

Use pointer for array

```
#include<stdio.h>

main()
{
    char a[6] = "ABCDEF";
    char* ptr1 = &a[0]; //first address
    char* ptr2 = &a[5]; //last address

    int length = 0;
    while(ptr1 < ptr2)
    {
        char temp = *ptr1;
        *ptr1 = *ptr2;
        *ptr2 = temp;

        ptr1++;
        ptr2--;
    }
    printf("Inversion is %s\n", a);
}
```

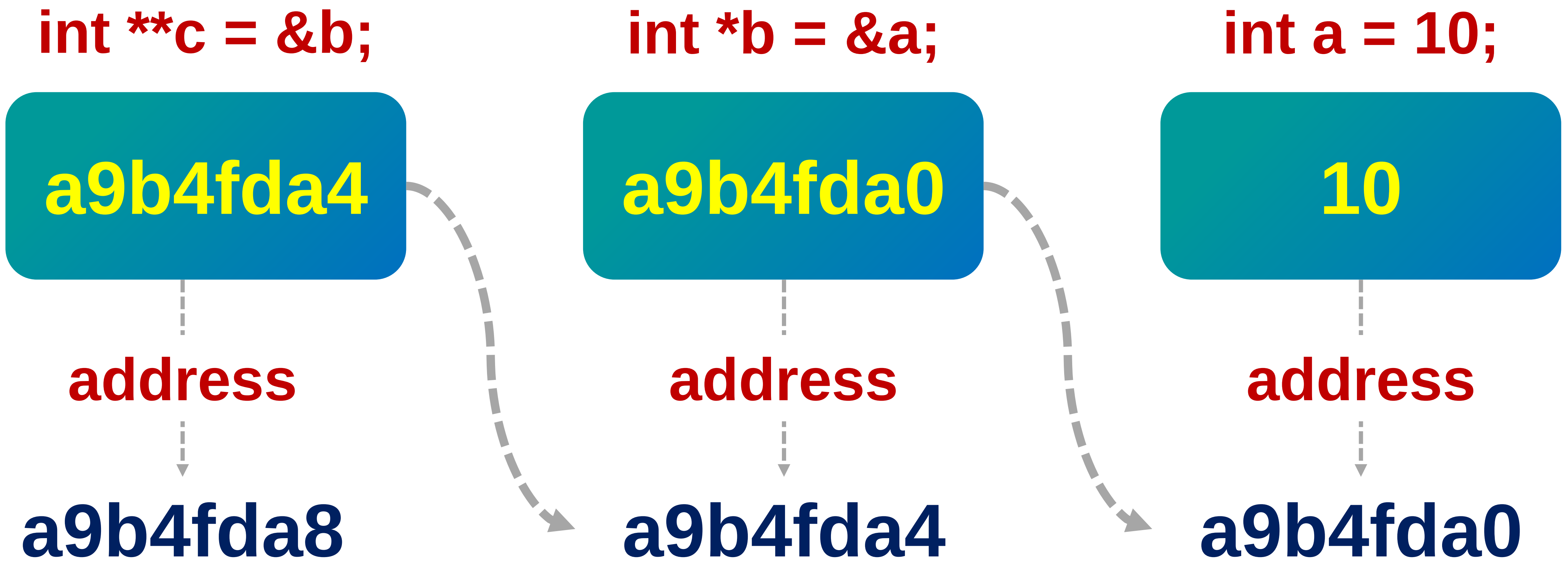
How to invert a string?



Inversion is FEDCBA

Double pointer

Pointer to pointer



Double pointer

Double pointer can represent matrix!

Single pointer

```c
main()
{
    int r = 3, c = 4;
    int* ptr = malloc((r * c) * sizeof(int));

    for (int i = 0; i < r * c; i++)
        ptr[i] = i + 1;

    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++)
            printf("%d ", ptr[i * c + j]);
        printf("\n");
    }
}
```

1	2	3	4	5	6	7	8	9	10	11	12

Double pointer

```c
main()
{
    int r = 3, c = 4;
    int** arr = (int**)malloc(r * sizeof(int));

    for (int i = 0; i < r; i++)
        arr[i] = (int*)malloc(c * sizeof(int));

    int count = 0;
    for (int i = 0; i < r; i++)
        for (int j = 0; j < c; j++)
            arr[i][j] = ++count;
}
```

1	2	3	4
5	6	7	8
9	10	11	12

NULL pointer

Always good to assign NULL to a pointer variable if no address is assigned.

```
#include <stdio.h>
main()
{
   int *ptr = NULL;
   printf("The address of ptr is : %x\n", &ptr);
   printf("The value of ptr is : %x\n", ptr); //0
}
```

Memory management

You can use this library to manage the **memory** of C program

```
#include <stdlib.h>
```

Memory management

C provides several functions for memory allocation and management.

function	Description
calloc(int num, int size)	Allocate an array of **num** elements each with **size (in byte)**
malloc(int num)	Allocate an array of num bytes and leave them uninitialized
realloc(void *addr, int newsize)	Re-allocate memory at **addr**ess with **newsize**
free(void *addr)	Release a block of memory at **addr**ess

calloc() & malloc()

calloc()

contiguous/连续的
allocation

↓

allocates memory and
initializes all bits to zero

malloc()

memory
allocation

↓

allocates memory and leaves
the memory uninitialized

calloc() function

Fixed array size, fixed memory

```
char name[100];
```

```
char *name;
name = (char*)calloc(200, sizeof(char));
```

Dynamic memory at address of name (200 bytes)

malloc() function

Fixed array size, fixed memory

```
char name[100];
```

```
char *name;
name = (char*)malloc(200*sizeof(char));
```

Dynamic memory at address of name (`200 bytes`)

calloc() & malloc()

allocates memory and initializes all bits to zero

```
char *name;

name = (char*)calloc(200, sizeof(char));

name = (char*)malloc(200*sizeof(char));
```

allocates memory and leaves the memory uninitialized

realloc() function

$$\begin{tikzcd}
 & \text{Increase memory} \\
\textbf{realloc()} \arrow[ur, bend left] \arrow[dr, bend right] & \\
 & \text{Decrease memory}
\end{tikzcd}$$

Repeatedly
allocation

$$\downarrow$$

Re-allocates memory to the
pointer variable

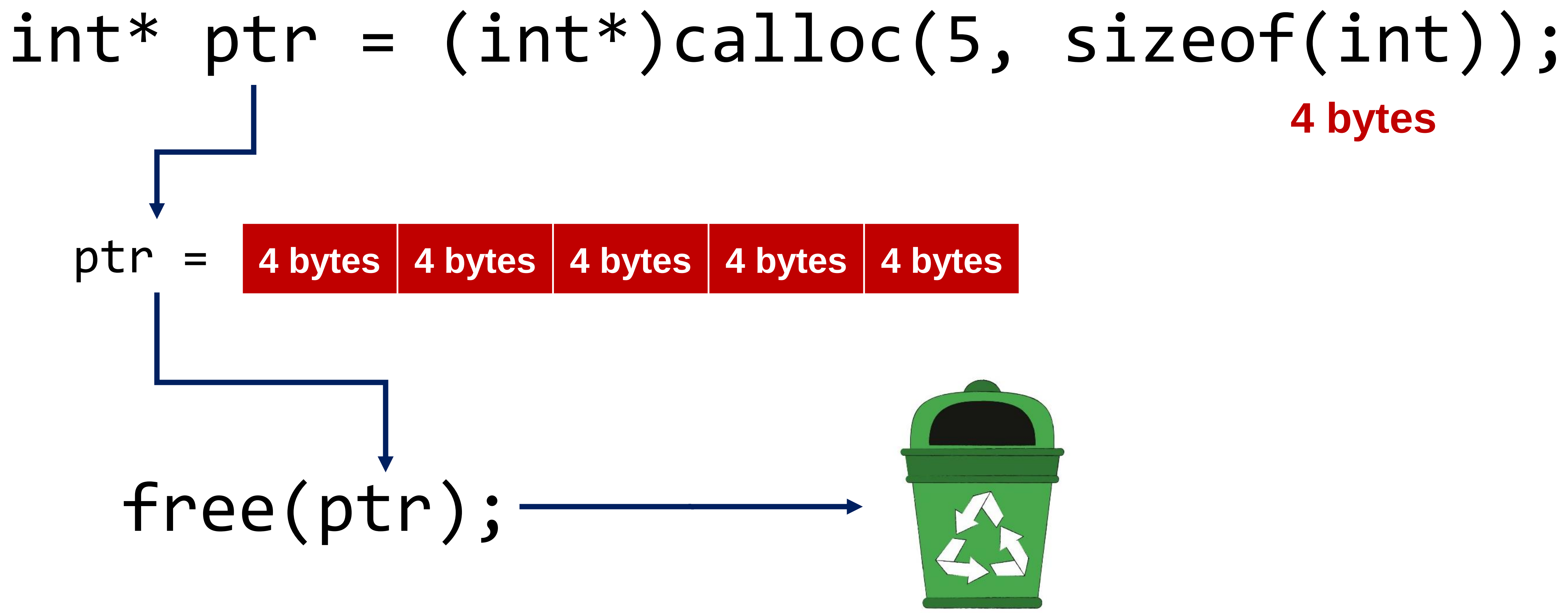
realloc() function

Allocate memory at address of name (200 bytes)

```
char *name;
name = (char*)malloc(200*sizeof(char));


name = (char*)realloc(name, 100*sizeof(char));
```

Resize the merry at address of name (100 bytes)

free() function



```c
int* ptr = (int*)calloc(5, sizeof(int));
```

```c
free(ptr);
```

Dynamic memory allocation

Use pointers as **output** of function to return results!

```
ptr[0] = v1 + v2;
ptr[1] = v1 - v2;
ptr[2] = v1 * v2;
ptr[3] = v1 / v2;
```

```
int* func(int v1, int v2)
{
    int* ptr = (int*) calloc(4, sizeof(int));
    ptr + 0 = v1 + v2;
    ptr + 1 = v1 - v2;
    ptr + 2 = v1 * v2;
    ptr + 3 = v1 / v2;
    return ptr;
}

main()
{
    int a = 10, b = 5;
    int *ptr = func(a, b);
    printf("sum=%d, sub=%d, mul=%d, div=%d", *ptr, *(ptr+1), *(ptr+2),
*(ptr+3)));
}
```

Output a pointer (array)

Dynamic memory allocation

Static array & Dynamic array

静态 动态

```
int main(void)
{
    char str1[] = "Ilove";

    char* str1_ = (char*)malloc(sizeof(char) * 6);

    for (int i = 0; i < strlen(str1) + 1; i++)
        str1_[i] = str1[i];

return 0;
}
```

ILove

We can convert static array to dynamic array

Summary

- **Pointer** is a variable that stores the address of another variable.
- We can access the **memory address** directly using the pointer.
- By changing the pointer value, the value stored at the address will be modified, typically useful for **functions** to pass values.
- Pointer can point to arrays, using **arithmetic and logical operations** (++, --, ==, >, <) to scan the memory address.
- We can **manage the memory** using C provided functions in **stdlib.h,** e.g. calloc(), malloc(), relloc(), free().

5 questions

1. What is the difference between the variable and pointer variable?

2. Which of following is the correct statement for a pointer ( )
A. int a = 5; int *p = &a;
B. char a = ‘a’; char *p = a;
C. int *p = 5;
D. Above are correct

3. Given int a[] = {1,2,3,4,5,6}, assume int *ptr = &a[2]; which of following is true( )
A. *(ptr+2) is 3
B. *(ptr+2) is 4
C. *(ptr+2) is 5
D. *(ptr+2) is 6

5 questions

4. Assuming the function is **void f(int, int*)**, in the main function, we have

```
int a  = 2;
int *p = &a;
```

Which of following function calling is correct?

A. f(a, &p)
B. f(*p, p)
C. f(p, a)
D. f(*p, a)

5. Assume A = 2, B = 10, write a function to swap the value between A and B (B = 2, A = 10) ?

Content

What data needs to be structured?

In real applications, an object needs to be described by different types of data

Patient

- ❑ Name
- ❑ Age
- ❑ Gender
- ❑ Disease
- ❑ Vital signs
- ❑ Medical records
- ❑ Symptoms

Three types of structure

Struct (结构体)

👍👍👍

Used very often

Union (共用体)

Useless

↓

Union defines a new data type that allows using variables with different types at the **same memory location**!

Enum (枚举型)

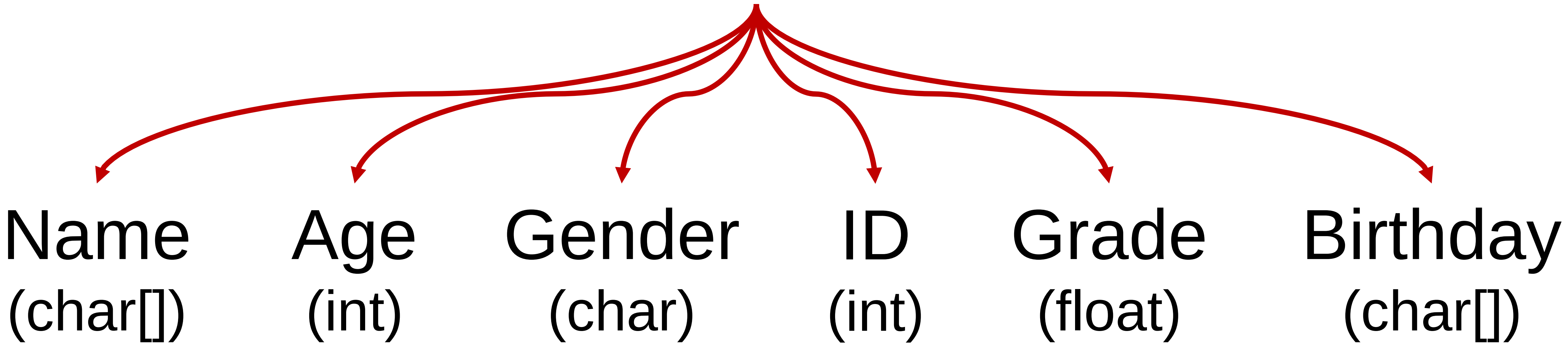
👍

Used but not often

Struct

You cannot use **array** to group data with different types

Struct (结构体)



What is struct?

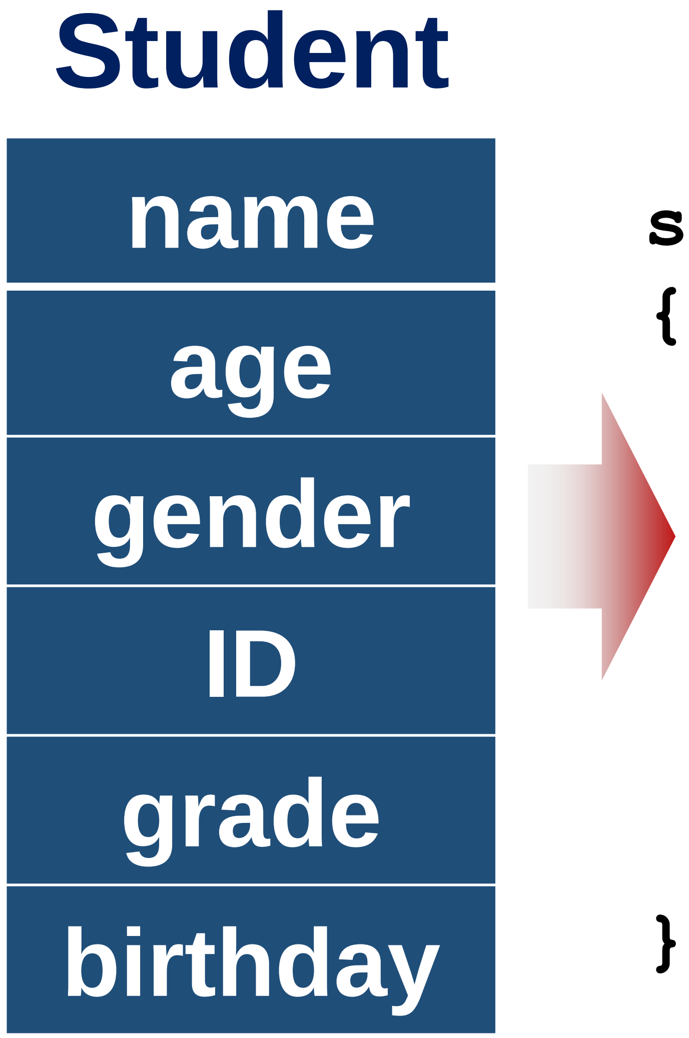
Struct defines a new data type that allows using variables with different types.

```
struct name
{
        type variable;
        type variable;
        type variable;
};
```

Struct name

Member list

How to define struct?



Struct name

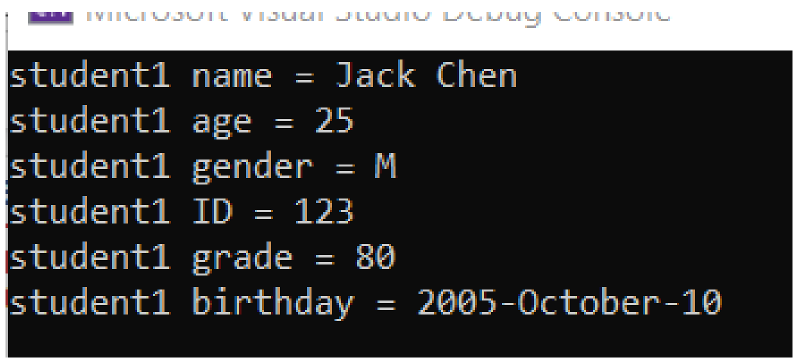
```
struct student
{
    char name[20];
    int age;
    char gender;
    int ID
    int grade;
    char birthday[50];
};
```

Member list

How to define struct?

```
#include<stdio.h>
struct student
{
    char name[20];
    int age;
    char gender;
    int ID;
    int grade;
    char birthday[50];
};

main()
{
    struct student student1 = {"Jack Chen", 25, 'M', 123, 80, "2005-October-10"};

    printf("student1 name = %s\n", student1.name);
    printf("student1 age = %d\n", student1.age);
    printf("student1 gender = %c\n", student1.gender);
    printf("student1 ID = %d\n", student1.ID);
    printf("student1 grade = %d\n", student1.grade);
    printf("student1 birthday = %s\n", student1.birthday);
}
```

Initialize the struct when declaring it, must be in order!



```
student1 name = Jack Chen
student1 age = 25
student1 gender = M
student1 ID = 123
student1 grade = 80
student1 birthday = 2005-October-10
```

How to define struct?

```
#include<stdio.h>
struct student
{
    char name[20];
    int age;
    char gender;
    int ID;
    int grade;
    char birthday[50];
};

main()
{
    struct student student1 = { "Jack Chen",    25, 'M', 123, 80, "2005-October-10" };
    struct student student2 = { "Li Wang",      23, 'F', 124, 87, "2004-May-9" };
    struct student student3 = { "Steffen He",   24, 'M', 125, 92, "2005-September-12" };
    struct student student4 = { "Tomas Huang",  25, 'M', 126, 90, "2005-March-23" };
    struct student student5 = { "Helen Luo",    26, 'F', 127, 84, "2005-February-15" };
}
```

Declare and define a group of students!

Array of structs

```
struct student
{
    char name[20];
    int ID;
    char gender;
};
```

① declare struct array and initialize it separately

```
main()
{
    struct student stu[2];
    strcpy(stu[0].name, "Jack");
    stu[0].ID = 1;
    stu[0].gender = 'M';

    strcpy(stu[1].name, "Merry");
    stu[1].ID = 2;
    stu[1].gender = 'F';
}
```

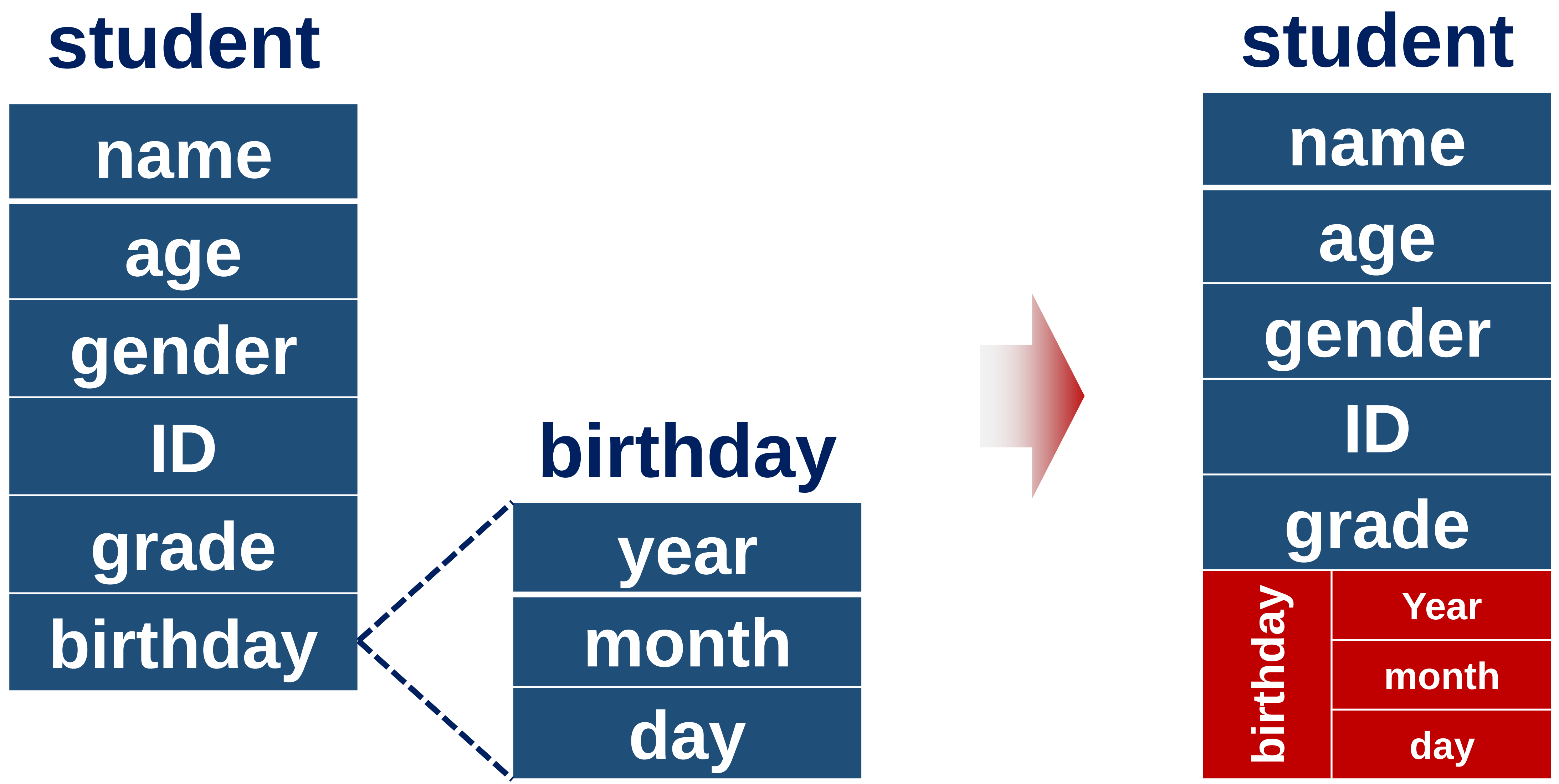
Array of structs

② declare struct array and initialize it jointly

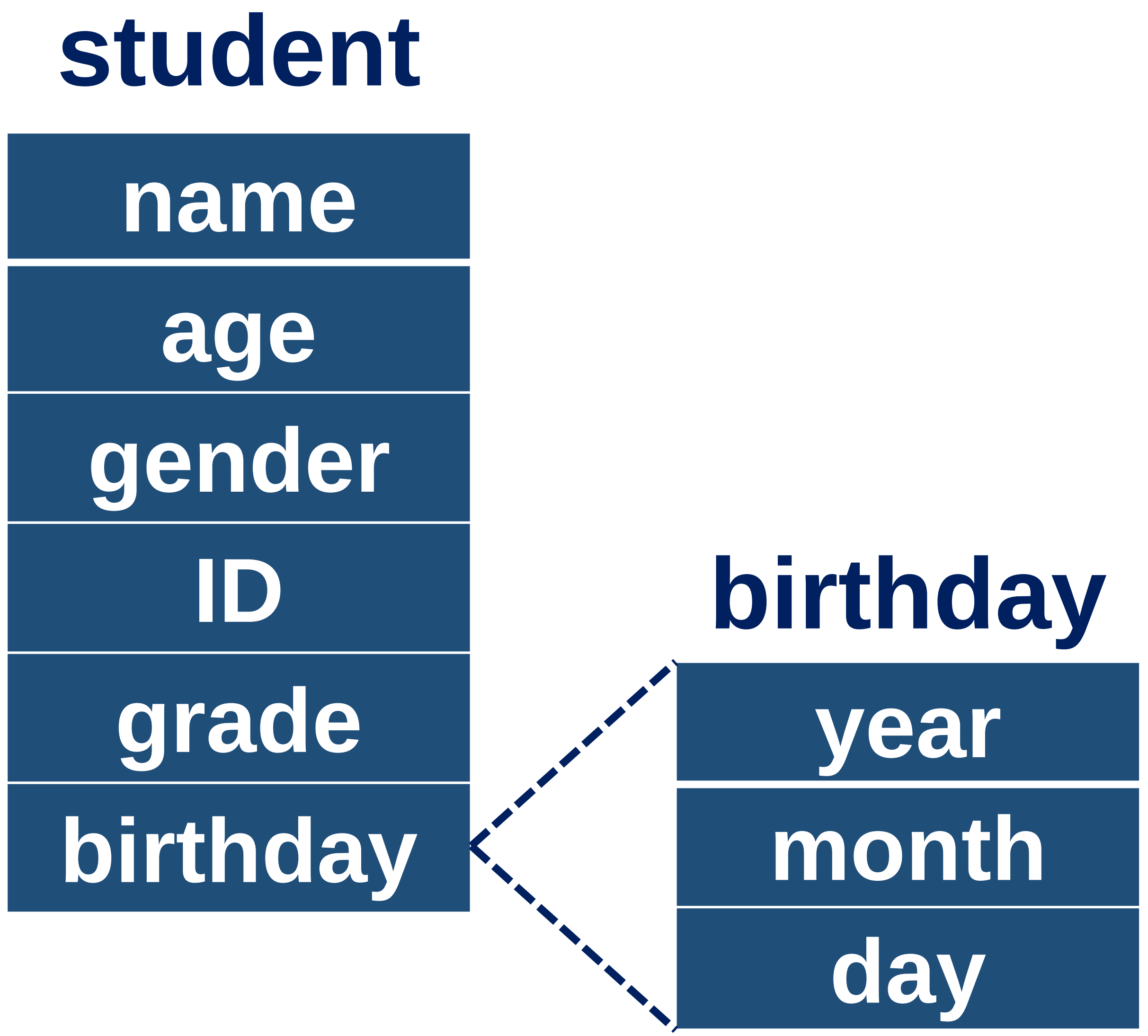
```
struct student
{
    char name[20];
    int ID;
    char gender;
};
```

```
main()
{
    struct student stu[2] = {
{"Jack",1,'M'}, {"Merry",2,'F'}};
}
```

Nested structs



Nested structs



```
struct birthday
{
    int year;
    int month;
    int day;
};

struct student
{
    char name[20];
    int age;
    char gender;
    int ID;
    int grade;
    struct birthday birth;
};
```

Nested structs

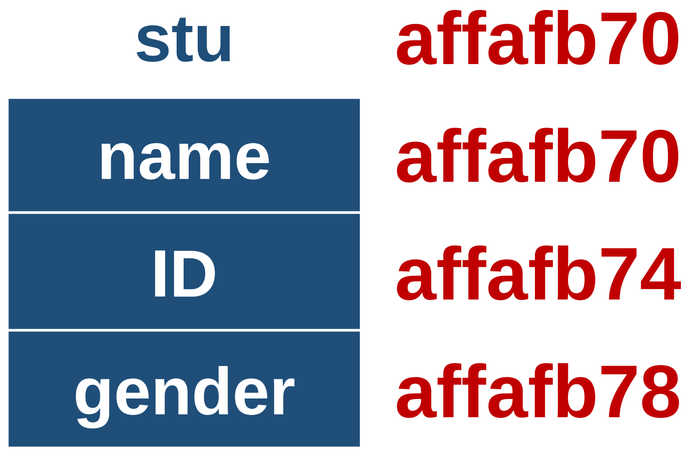
```c
#include<stdio.h>

struct birthday
{
    int year;
    int month;
    int day;
};

struct student
{
    char name[20];
    int age;
    char gender;
    int ID;
    int grade;
    struct birthday birth;
};
```

```c
main()
{
    struct student student1;
    strcpy(student1.name, "Jack Chen");
    student1.age = 25;
    student1.gender = 'M';
    student1.ID = 123;
    student1.grade = 80;
    student1.birth.year = 2005;
    student1.birth.month = 10;
    student1.birth.day = 10;

    printf("student1 name = %s\n", student1.name);
    printf("student1 age = %d\n", student1.age);
    printf("student1 gender = %c\n", student1.gender);
    printf("student1 ID = %d\n", student1.ID);
    printf("student1 grade = %d\n", student1.grade);
    printf("student1 birthday = %d-%d-%d\n", 
student1.birth.year, student1.birth.month, student1.birth.day);
}
```

Pointer to structs

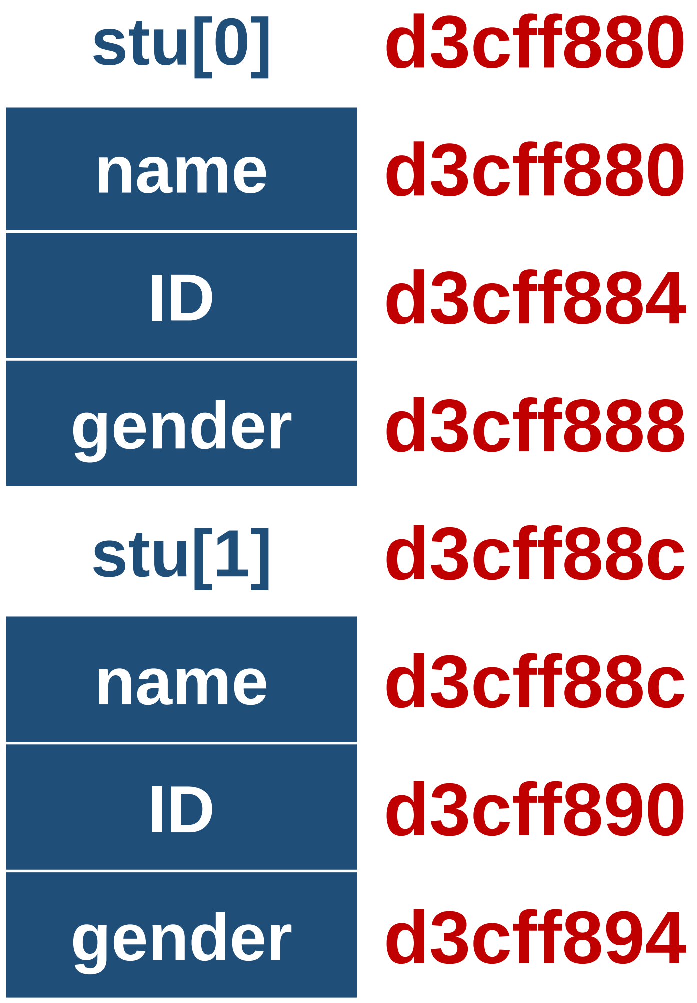
```
#include<stdio.h>

struct student
{
    char name[4];
    int ID;
    char gender;
};

main()
{
    struct student stu = {"Jack",  1, 'M'};

    printf("Address of stu: %x\n", &stu);
    printf("Address of num: %x\n", &stu.name);
    printf("Address of ID: %x\n", &stu.ID);
    printf("Address of gender: %x\n", &stu.gender);
}
```

You can check the address of struct!!!



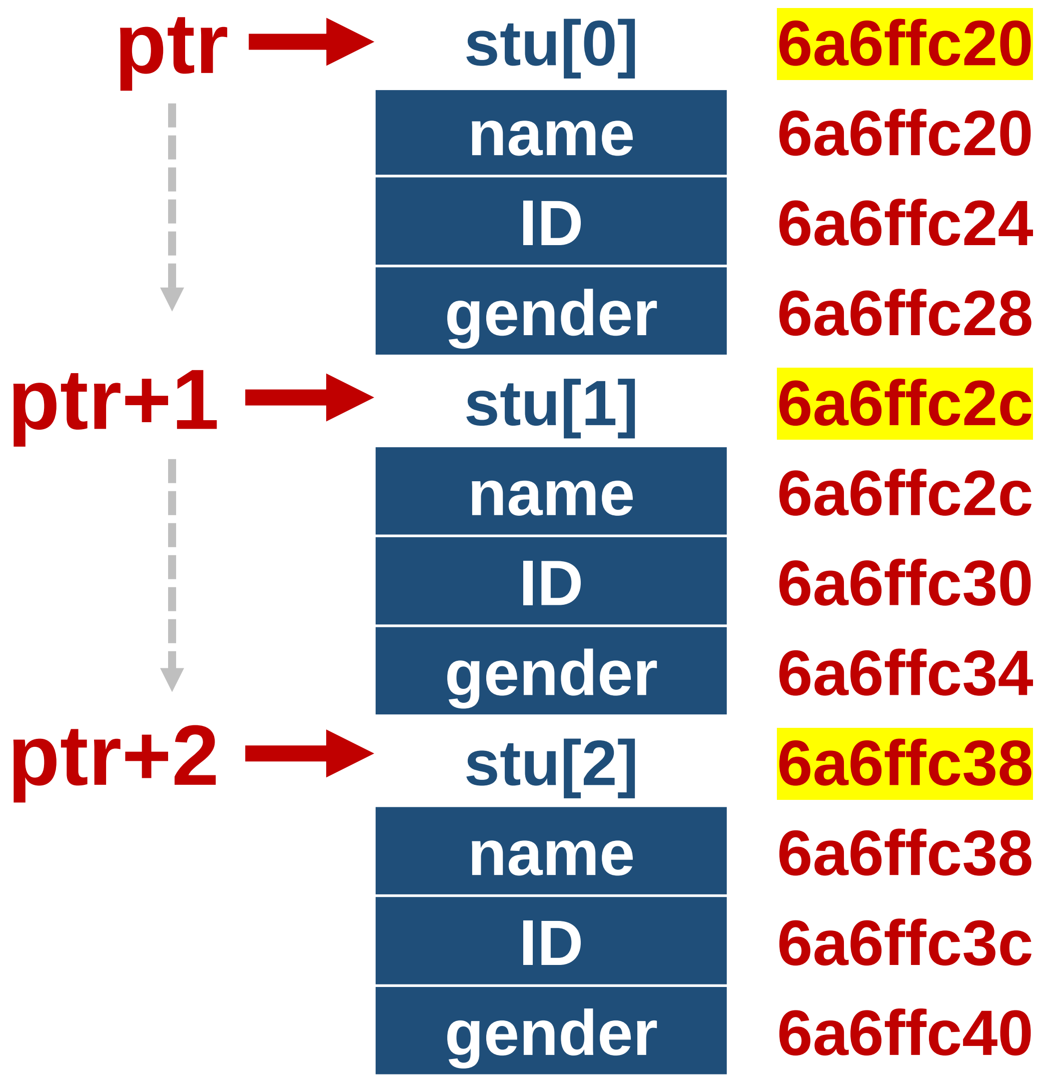
Pointer to structs

```c
#include<stdio.h>

struct student
{
    char name[4];
    int ID;
    char gender;
};

main()
{
    struct student stu[2] = {{"Jack",  1, 'M'}, {"Jen",  1, 'F'}};

    for (int i = 0; i < 2; i ++)
    {
        printf("Address of stu: %x\n", &stu[i]);
        printf("Address of num: %x\n", &stu[i].name);
        printf("Address of ID: %x\n", &stu[i].ID);
        printf("Address of gender: %x\n", &stu[i].gender);
    }
}
```



Pointer to structs

```c
#include<stdio.h>

struct student
{
    char name[4];
    int ID;
    char gender;
};

main()
{
    struct student stu[3] = {{"Jack",  1, 'M'},
{"Jen",  2, 'F'}, {"Mike", 3, "M"}};

    struct student* ptr = stu; //&stu[0]

    printf("address of ptr = %x\n", ptr);
    printf("address of ptr+1 = %x\n", ptr+1);
    printf("address of ptr+2 = %x\n", ptr+2);
}
```



Pointer to structs

```c
#include<stdio.h>
struct student
{
    char name[4];
    int ID;
    char gender;
};
main()
{
    struct student stu[3] = { {"Jack",  1, 'M'},
{"Jen",  2, 'F'}, {"Mike", 3, "M"} };

    struct student* ptr = stu; //&stu[0];

    printf("stu 1 name = %s\n", (*ptr).name);
    printf("stu 2 name = %s\n", (*(ptr + 1)).name);
    printf("stu 3 name = %s\n", (*(ptr + 2)).name);

    printf("stu 1 name = %s\n", ptr->name);
    printf("stu 2 name = %s\n", (ptr + 1)->name);
    printf("stu 3 name = %s\n", (ptr + 2)->name);
}
```

How to access members of struct using pointer?

①
```c
(*ptr).name
(*ptr).ID
(*ptr).gender
```

②
```c
ptr->name
ptr->ID
ptr->gender
```

Struct for functions

Struct as input parameters for function

Struct as output results of function

Struct as function input & output

Struct pointer

```
#include<stdio.h>
struct student
{
    char name[5];
    int ID;
    char gender;
};
input(struct student *stu);

main()
{
    struct student stu = { "Jack",  1, 'M' };
    input(&stu);
    printf("%s - %d - %c", stu.name, stu.ID,
stu.gender);
}

input(struct student *stu)
{
    strcpy(stu->name, "Lily");
    stu->ID = 5;
    stu->gender = 'F';
}
```

Return struct

```
#include<stdio.h>
struct student
{
    char name[5];
    int ID;
    char gender;
};
struct student get();

main()
{
    struct student stu = get();
    printf("%s - %d - %c\n", stu.name, stu.ID,
stu.gender);
}

struct student get()
{
    struct student stu;
    strcpy(stu.name, "Lily"); stu.ID = 5; stu.gender =
'F';
    return stu;
}
```

Enumerate

Enum is a user defined data type in C, **assign names to integer constants**, for a program easy to read and maintain.

```
enum [union tag]
{
    variable;
    variable;
     …
};
```

← **All integers by default!**

Enumerate

```
enum week { Mon, Tue, Wed, Thu, Fri, Sat, Sun };
             0    1    2    3    4    5    6

enum week { Mon=1, Tue, Wed, Thu, Fri, Sat, Sun };
              1     2    3    4    5    6    7


enum week day;
day = Mon;
```

Enumerate

Enum assigns names to integer constants

```c
#include<stdio.h>

enum week { Mon, Tue, Wed, Thur, Fri,
Sat, Sun };

main()
{
    for (int i = Mon; i <= Sun; i++)
    {
        printf("%d\n", i);
    }
}
```

```c
#include<stdio.h>

enum Year { Jan, Feb, Mar, Apr, May,
June, July, Aug, Sept, Oct, Nov, Dec };

main()
{
    for (int i = Jan; i <= Dec; i++)
    {
        printf("%d\n", i);
    }
}
```

Typedef

Typedef allows you to create a type with new name.

```
Typedef type name;
```

```
typedef unsigned char BYTE;
```

```
typedef struct Books
{
    char title[50];
    char author[50];
    int book_id;
} Book;
```

Use typedef for variable

```
unsigned char var1;
```

```
typedef unsigned char BYTE;

BYTE var1;
```

A new type

```
struct Books
{
    char title[50];
    char author[50];
    char subject[100];
    int book_id;
};

struct Books book1;
```

```
typedef struct Books
{
    char title[50];
    char author[50];
    char subject[100];
    int book_id;
} Book;

Book book1;
```

A new type

#define

#define allows you to define **macros** (constant values) as pre-processors before compilation.

```
#define name value
```

Macro (宏) definitions must be constant, there is no symbols like = and ;

#define

#define sets macros for numbers, strings or expressions

numbers

```
#define ONE 1
#define PI 3.14
```

strings

```
#define CHAR 'a'
#define NAME "SUSTech"
```

Expr.

```
#define MIN(a, b) (a < b ? a : b)
#define SUM(a, b, c) (a + b + c)
```

#define

Use const global variable

```c
#include<stdio.h>

const int ONE = 1;
const float PI = 3.14;
const char CHAR = 'a';
const char NAME[] = "SUSTech";

main()
{
    printf("%d\n", ONE);
    printf("%f\n", PI);
    printf("%c\n", CHAR);
    printf("%s\n", NAME);
}
```

Use #define macro

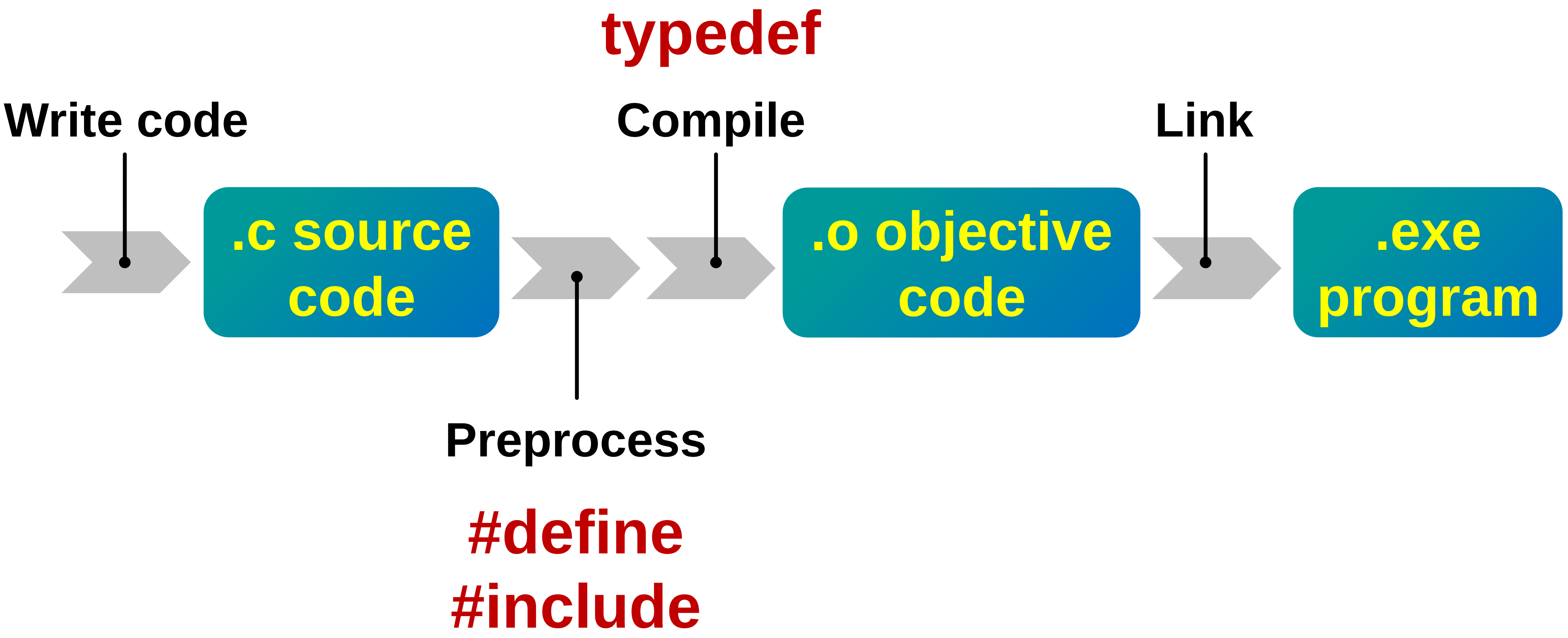
```c
#include<stdio.h>

#define ONE 1
#define PI 3.14
#define CHAR 'a'
#define NAME "SUSTech"

main()
{
    printf("%d\n", ONE);
    printf("%f\n", PI);
    printf("%c\n", CHAR);
    printf("%s\n", NAME);
}
```

#define

```
#include<stdio.h>

#define MIN(a, b) (a < b ? a : b)
#define SUM(a, b, c) (a + b + c)
#define POW(x) (x * x)

main()
{
    printf("%d\n", MIN(10, 100));
    printf("%d\n", SUM(10, 20, 30));
    printf("%f\n", SUM(2.3, 0.8, -3.5));
    printf(“%d\n”, POW(5));
}
```

$\}$ **Use #define for macro expressions**

typedef versus #define



typedef versus #define

C source code

```
#include<stdio.h>

#define SQUARE(X) X*X
#define PR(x) printf("The
result is %d.\n",x)
int main()
{
    int z = 5;
    PR(SQUARE(z));
    PR(SQUARE(z + 2));
    PR(100 / SQUARE(2));
    return 0;
}
```

Preprocessed code

```
int main()
{
    int z = 5;
    printf("The result is %d.\n",
z*z);
    printf("The result is %d.\n",
(z + 2)*(z + 2));
    printf("The result is
%d.\n",100/(2*2));
    return 0;
}
```

typedef versus #define

typedef

- Processed by **compiler**, actual definition of a new type
- Give symbolic names to types
- With scope rules. If defined inside function, only visible to the function

#define

- Processed by **preprocessor**, copy-paste the definition in place
- Define alias for values (#define ONE 1)
- No scope rules, it replaces all occurrences, visual everywhere

Summary

- **Struct** can be used to define a new data type for grouping data with different types.
- Struct is very useful and has been commonly used. It can be used with **arrays, pointers, and functions**.
- **Union** is not useful (only need know). **Enum** can be used to assign a sequence of names with integers.
- **Typedef** can define a new type of data by combining existing ones (short int, struct), **#define** can define marcos for pre-processing.

5 questions

1. Which of following keyword can define a struct? ( )
A. union B. enum C. struct D. typedef

2. What is the difference between struct (结构体) and union (共用体)? Which one is suggested to use if want to store data with multiple types at the same time?

3. Which of following statement is correct to access the member of struct on the right? ( )

A. (*p).data.a B. (*p).a C. *p->a D. p.data

```
typedef struct Data {
    int a;
    float b;
}data;

int main()
{
    data d = {5, 3.14};
    data *ptr = &d;
}
```

5 questions

4. Which of following statement for marco definition? ( )

A. #define MIN(a, b) (a < b ? a : b)
B. #define POW(x) x * x
C. #define CHAR "SUSTech"
D. All above are correct

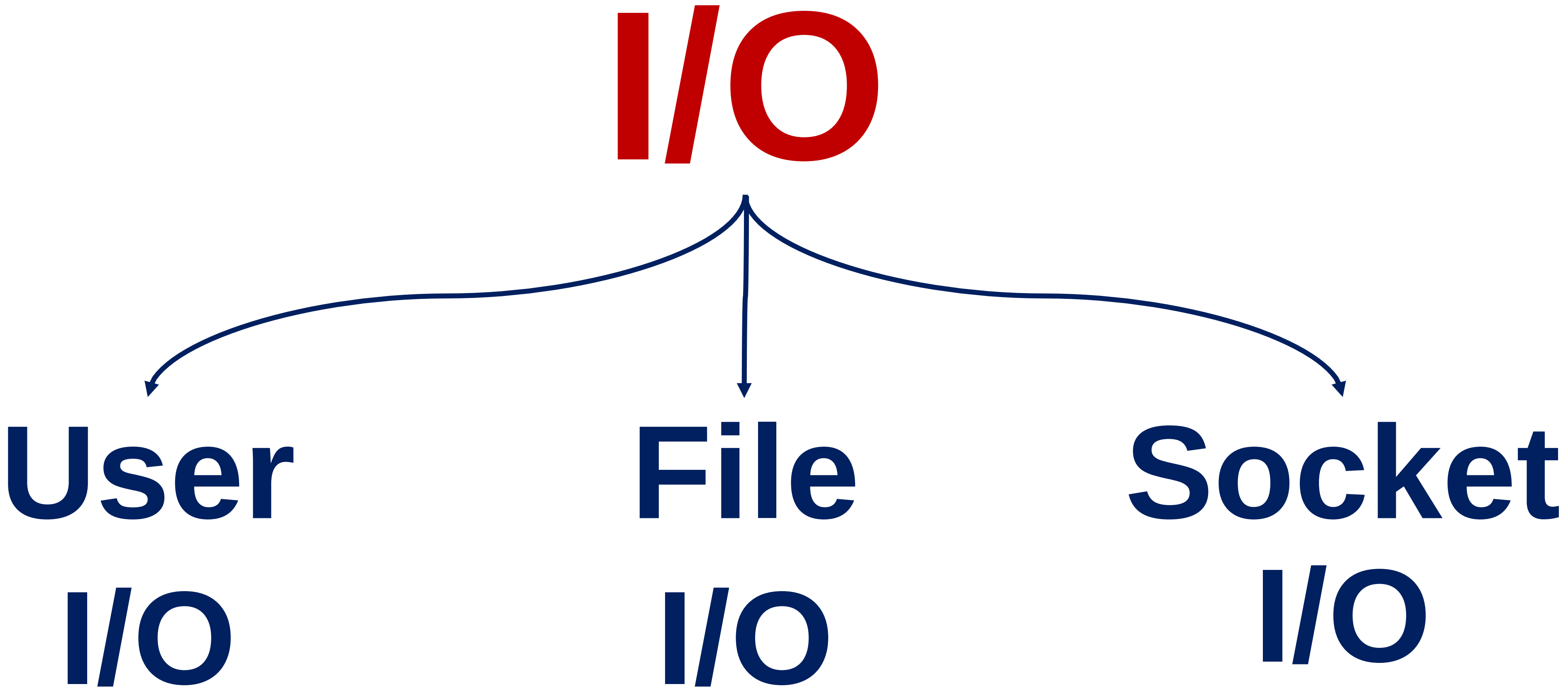
5. What is the value of fri?( )

A. 4 B.5 C. 9 D.10

```
enum week
{
    mon = 5, tue, wed, thr, fri, sat, sun
};
```

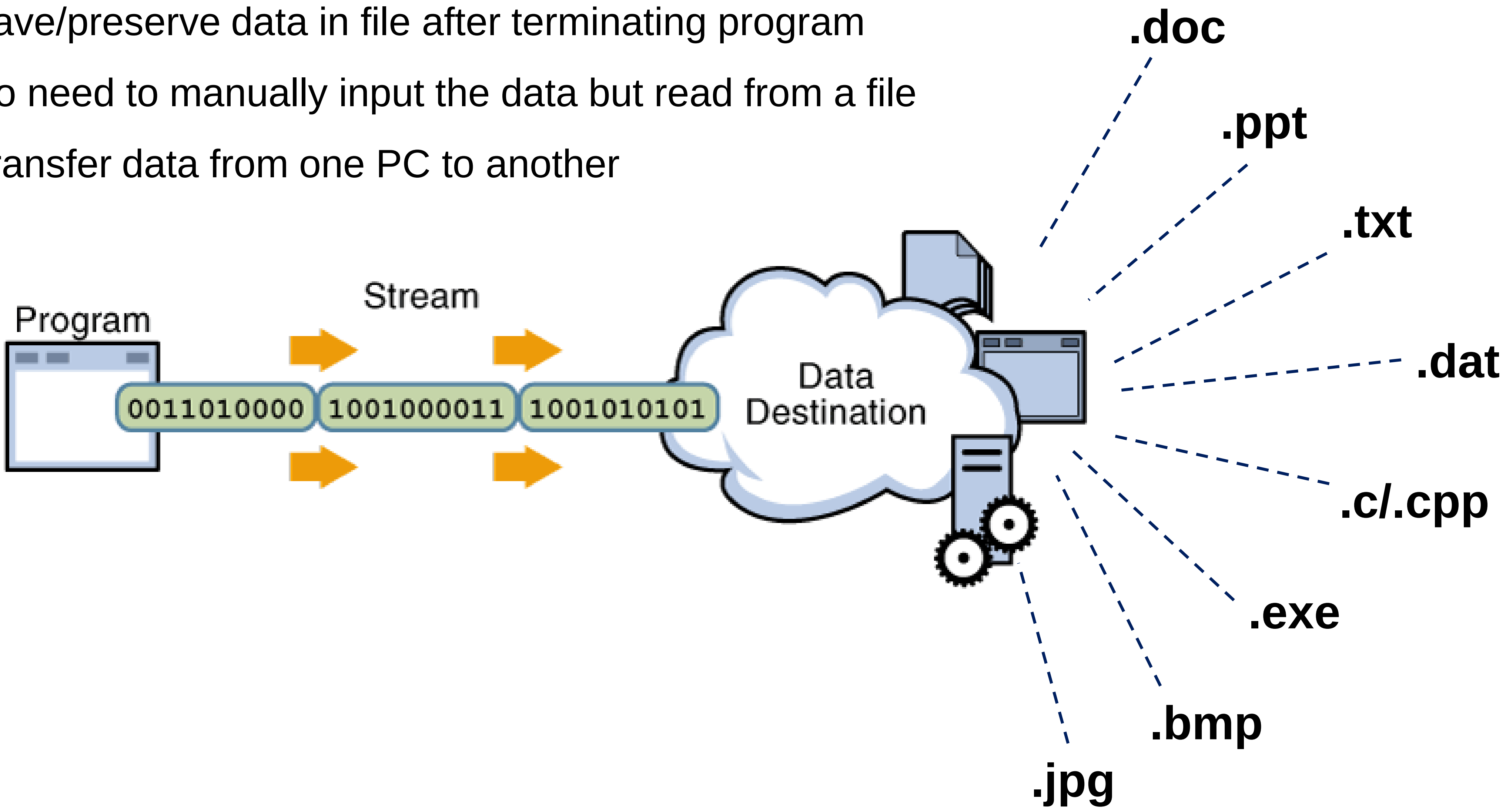
Content

Three types of I/O



File I/O

- Save/preserve data in file after terminating program
- No need to manually input the data but read from a file
- Transfer data from one PC to another



File formats

ASCII file

.txt

Plain **text**
(data in characters)

data.txt

.csv

Comma **S**eparated **V**alues
(data structured by “,”)

data.csv

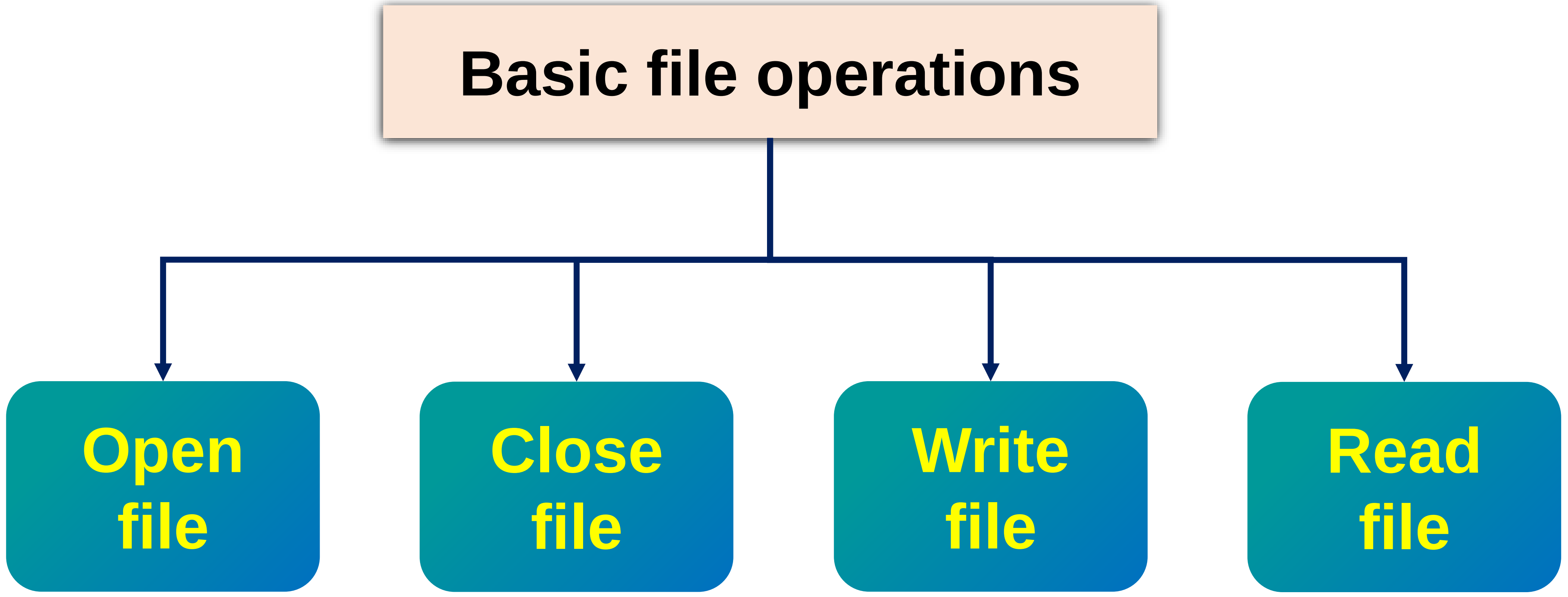
binary file

.bin

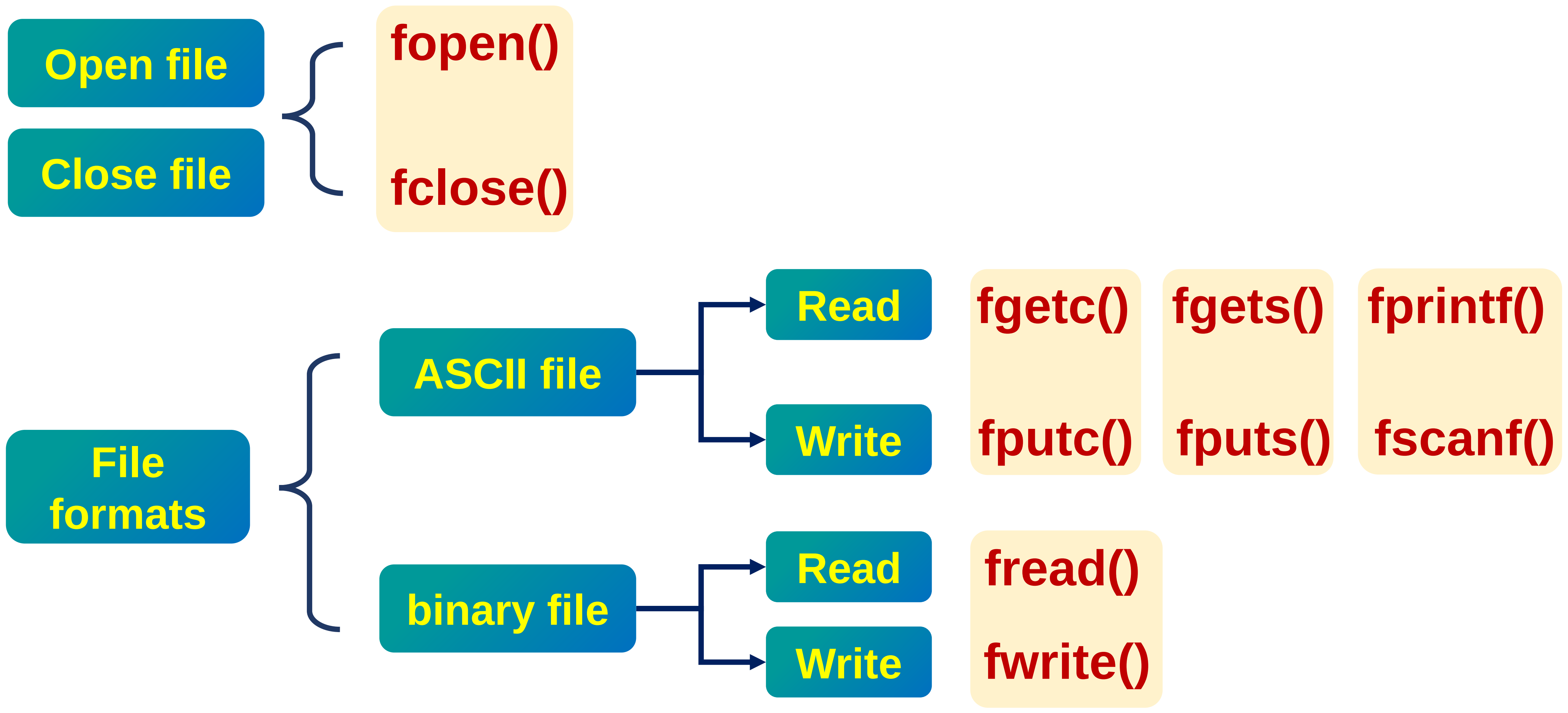
Binary values
(data in 0 and 1)

data.bin

File I/O functions



File I/O functions



Open file

Declare a FILE pointer (FILE belongs to #include<stdio.h>)

```c
FILE* fp;

fp = fopen(const char* filename, const char* mode);
```

Path of the file in the system

Absolute path
（绝对路径）

Relative path
（相对路径）

Mode (模式)

read

write

append

Open file: how to define path?

File name

File type

```
"data.txt"
```

```
"C:\\Users\\wenji\\Desktop\\c files\\data.txt"
```

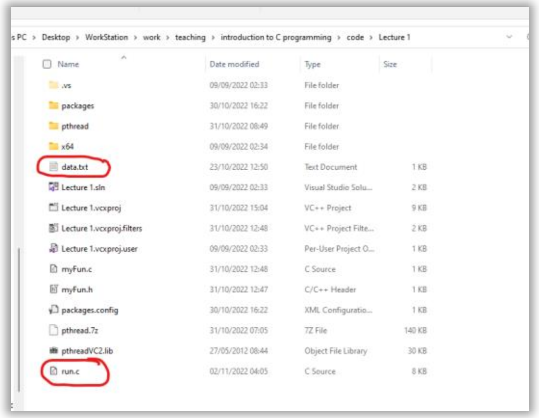
File path

Open file: how to define path?

```
FILE* fp;
```

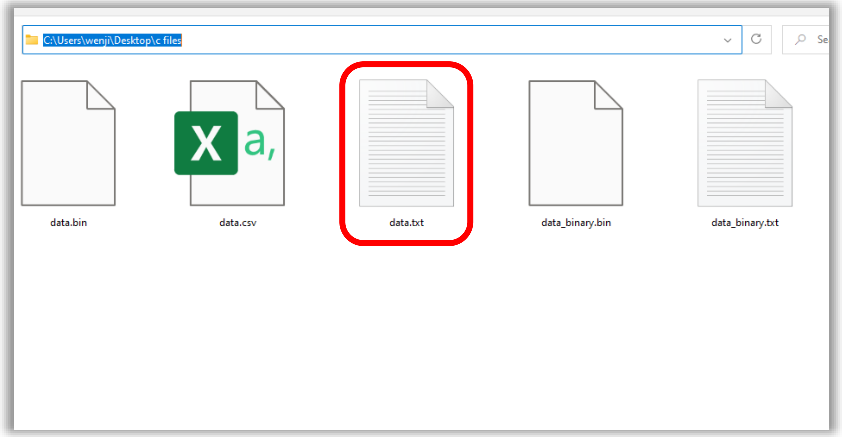
当前. c文件的路径（相对路径）

```
fp = fopen("test.txt", "w");
```


s PC › Desktop › WorkStation › work › teaching › introduction to C programming › code › Lecture 1

Name	Date modified	Type	Size
.vs	09/09/2022 02:33	File folder	
packages	30/10/2022 16:22	File folder	
pthread	31/10/2022 08:49	File folder	
x64	09/09/2022 02:34	File folder	
data.txt	23/10/2022 12:50	Text Document	1 KB
Lecture 1.sln	09/09/2022 02:33	Visual Studio Solu...	2 KB
Lecture 1.vcxproj	31/10/2022 15:04	VC++ Project	9 KB
Lecture 1.vcxproj.filters	31/10/2022 12:48	VC++ Project Filte...	2 KB
Lecture 1.vcxproj.user	09/09/2022 02:33	Per-User Project O...	1 KB
myFun.c	31/10/2022 12:48	C Source	1 KB
myFun.h	31/10/2022 12:47	C/C++ Header	1 KB
packages.config	30/10/2022 16:22	XML Configuratio...	1 KB
pthread.7z	31/10/2022 07:05	7Z File	140 KB
pthreadVC2.lib	27/05/2012 08:44	Object File Library	30 KB
run.c	02/11/2022 04:05	C Source	8 KB

```
fp = fopen("C:\\Users\\wenji\\Desktop\\c
files\\data.txt", "w");
```


C:\Users\wenji\Desktop\c files

系统的绝对路径

Open file: how to define mode?

r = read   w = write   a = append   b = binary

To open ASCII files (.txt, .csv):

"r", "w", "a", "r+", "w+", "a+" → 混合型

To open binary file (.bin):

"rb", "wb", "ab", "rb+", "r+b", "wb+", "w+b", "ab+", "a+b"

Close file

Input the FILE pointer

```
FILE *fp;
```

```
int fclose(FILE * fp);
```

- ✓ Flushes data pending in the buffer to file
- ✓ Closes the file
- ✓ Releases memory used for the file

Open and close a file

```
#include <stdio.h>
main()
{
    FILE* fp;

    fp = fopen("test.txt", "w+");

    // …

    fclose(fp);
}
```

Can be .txt, .bin, .csv

In a pair

Write\read a file

Write/read a single character:

```
int fputc(int c, FILE *fp);
int fgetc(FILE *fp);
```

Write/read a group of characters:

```
fputs(const char *s, FILE *fp);
fgets(char *buf, int n, FILE *fp);
```

Write/read formatted characters:

```
fprintf(FILE *fp, const char *format, ...);
fscanf(FILE *fp, const char *format, ...);
```

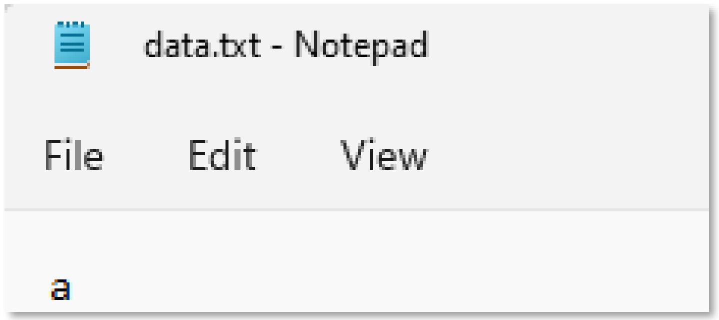
Write/read binary file:

```
fwrite(const void *ptr, int size_of_elements, int number_of_elements, FILE *fp);
fread(void *ptr, int size_of_elements, int number_of_elements, FILE *fp);
```

Write/read single character

Write

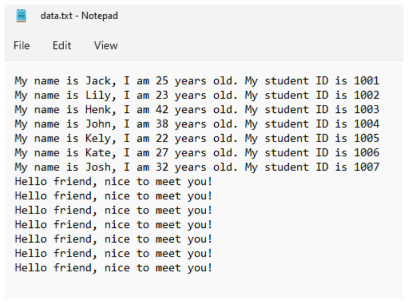
```c
#include <stdio.h>

main()
{
    FILE* fptr;
    fptr = fopen("data.txt", "w+");
    char data = 'a';
    fputc(data, fptr);

    fclose(fptr);
}
```



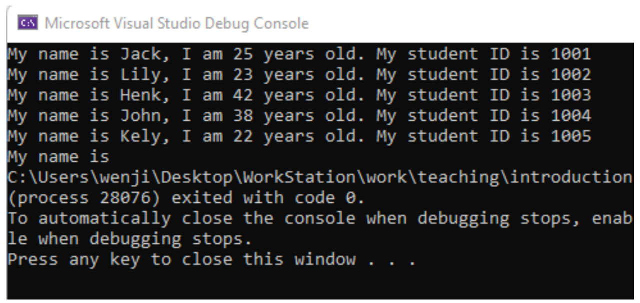
```
a
```

Read

```c
#include<stdio.h>

main()
{
    FILE* fptr;
    fptr = fopen("data.txt", "r");

    for (int i = 0; i < 300; i++)
    {
        char c = fgetc(fptr);
        printf("%c", c);
    }
    fclose(fptr);
}
```



```
My name is Jack, I am 25 years old. My student ID is 1001
My name is Lily, I am 23 years old. My student ID is 1002
My name is Henk, I am 42 years old. My student ID is 1003
My name is John, I am 38 years old. My student ID is 1004
My name is Kely, I am 22 years old. My student ID is 1005
My name is Kate, I am 27 years old. My student ID is 1006
My name is Josh, I am 32 years old. My student ID is 1007
Hello friend, nice to meet you!
Hello friend, nice to meet you!
Hello friend, nice to meet you!
Hello friend, nice to meet you!
Hello friend, nice to meet you!
Hello friend, nice to meet you!
Hello friend, nice to meet you!
```

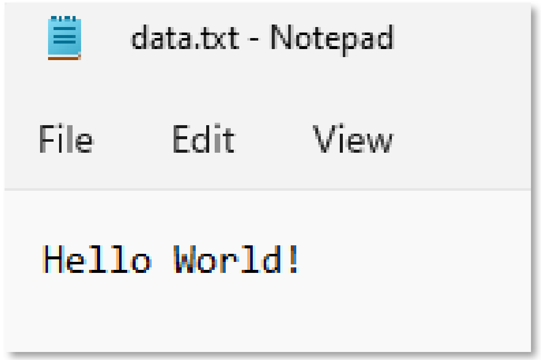


```
My name is Jack, I am 25 years old. My student ID is 1001
My name is Lily, I am 23 years old. My student ID is 1002
My name is Henk, I am 42 years old. My student ID is 1003
My name is John, I am 38 years old. My student ID is 1004
My name is Kely, I am 22 years old. My student ID is 1005
My name is
C:\Users\wenji\Desktop\WorkStation\work\teaching\introduction
(process 28076) exited with code 0.
To automatically close the console when debugging stops, enab
le when debugging stops.
Press any key to close this window . . .
```

Write/read a group of characters

Write

```c
#include<stdio.h>
main()
{
    FILE* fptr;
    fptr = fopen("data.txt", "w");

    char data[] = "Hello World!";
    fputs(data, fptr);

    fclose(fptr);

}
```

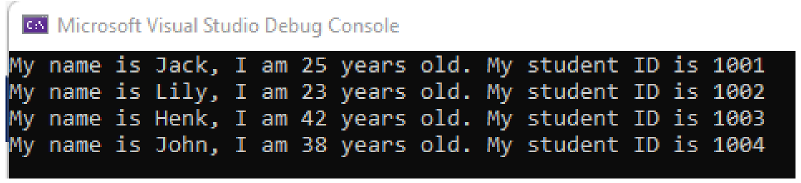


```
Hello World!
```

Read

```c
#include<stdio.h>

main()
{
    FILE* fptr;
    fptr = fopen("data.txt", "r");

    char data[300];

    fgets(data, 100, fptr); printf("%s", data);

    fgets(data, 100, fptr); printf("%s", data);

    fgets(data, 100, fptr); printf("%s", data);

    fgets(data, 100, fptr); printf("%s", data);

    fclose(fptr);
}
```

Length of string (N-1, last is null)



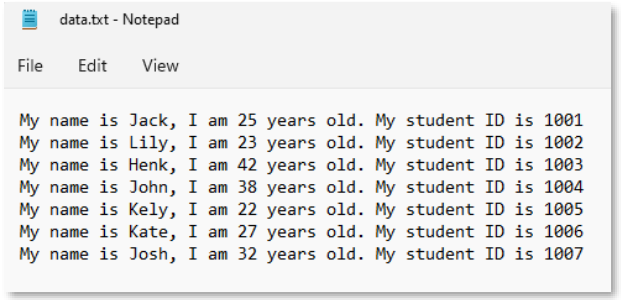
```
My name is Jack, I am 25 years old. My student ID is 1001
My name is Lily, I am 23 years old. My student ID is 1002
My name is Henk, I am 42 years old. My student ID is 1003
My name is John, I am 38 years old. My student ID is 1004
```

Write/read formatted characters

Write

```
#include<stdio.h>

main()
{
    FILE* fptr;
    fptr = fopen("data.txt", "w");

    char format[] = "My name is %s, I am %d
years old. My student ID is %d\n";

    fprintf(fptr, format, "Jack", 25, 1001);
    fprintf(fptr, format, "Lily", 23, 1002);
    fprintf(fptr, format, "Henk", 42, 1003);
    fprintf(fptr, format, "John", 38, 1004);
    fprintf(fptr, format, "Kely", 22, 1005);
    fprintf(fptr, format, "Kate", 27, 1006);
    fprintf(fptr, format, "Josh", 32, 1007);

    fclose(fptr);
}
```

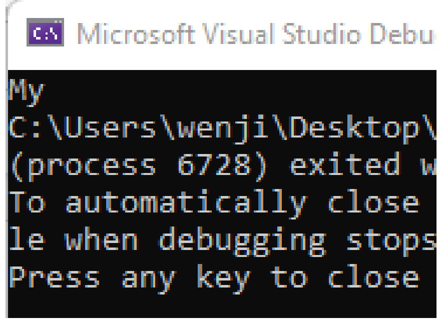
Writing mode



```
My name is Jack, I am 25 years old. My student ID is 1001
My name is Lily, I am 23 years old. My student ID is 1002
My name is Henk, I am 42 years old. My student ID is 1003
My name is John, I am 38 years old. My student ID is 1004
My name is Kely, I am 22 years old. My student ID is 1005
My name is Kate, I am 27 years old. My student ID is 1006
My name is Josh, I am 32 years old. My student ID is 1007
```

Read

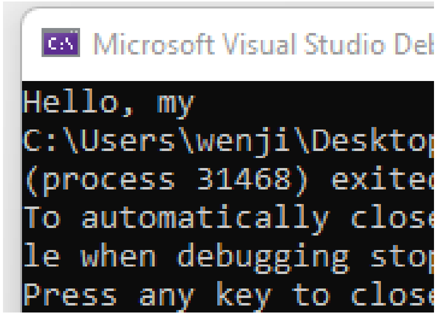
```
#include<stdio.h>

main()
{
    FILE* fptr;
    fptr = fopen("data.txt", "r");

    char str[20];
    fscanf(fptr, "%s", str);

    printf("%s", str);

    fclose(fptr);
}
```



```
My
C:\Users\wenji\Desktop\
(process 6728) exited w
To automatically close
le when debugging stops
Press any key to close
```

Read single word

```
#include<stdio.h>

main()
{
    FILE* fptr;
    fptr = fopen("data.txt", "r");

    char str1[20], str2[20];
    fscanf(fptr, "%s %s", str1, str2);

    printf("%s, %s", str1, str2);

    fclose(fptr);
}
```



```
Hello, my
C:\Users\wenji\Desktop
(process 31468) exited
To automatically close
le when debugging stop
Press any key to close
```

Read 2 words

Write/read binary file

Write

```c
#include<stdio.h>

typedef struct student {
    char name[100];
    int id;
    float average;
}stud;

main()
{
    FILE* fptr;
    fptr = fopen("data.bin", "wb");

    stud data[] = { {"Kate", 1001, 90},
{"Jack", 1002, 94}, {"Mike", 1003, 85} };
    fwrite(data, sizeof(data), 1, fptr);

    fclose(fptr);
}
```

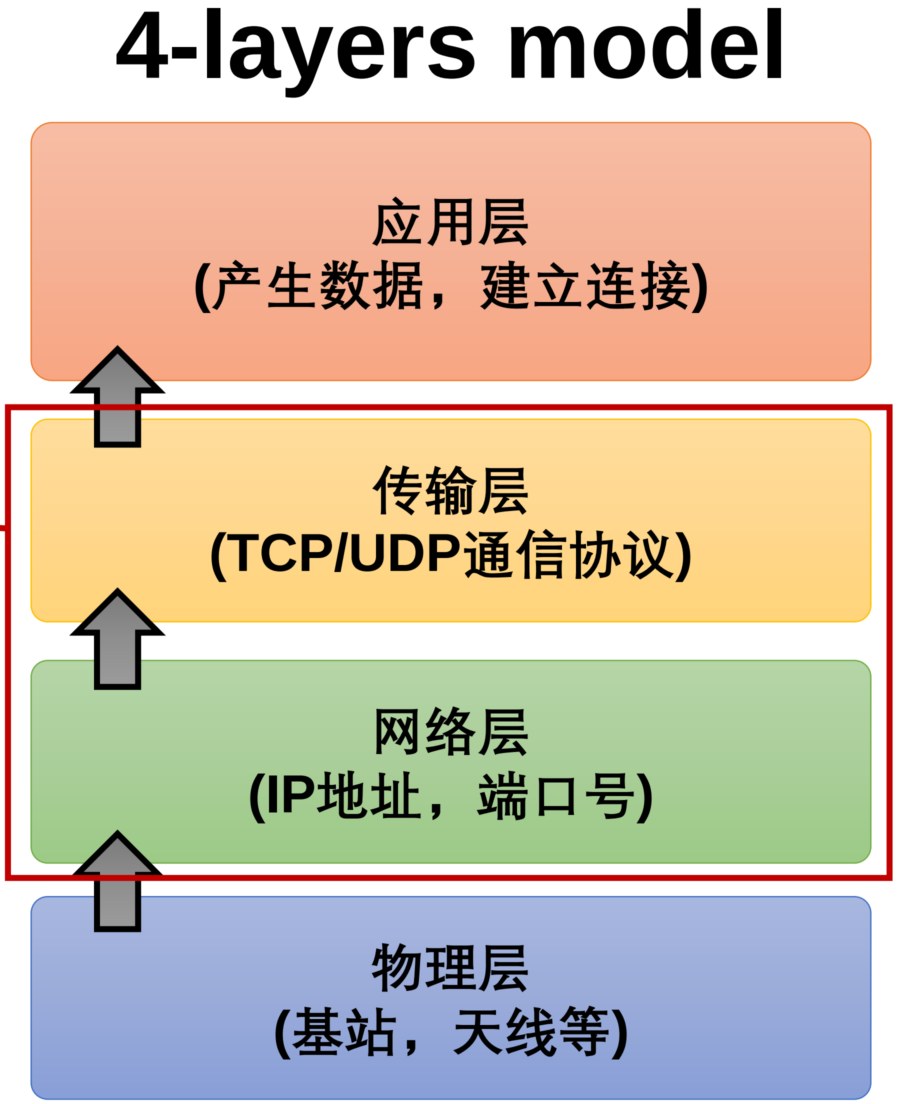
Read

```c
#include<stdio.h>

typedef struct student {
    char name[100];
    int id;
    float average;
}stud;

main()
{
    FILE* fptr;
    fptr = fopen("data.bin", "rb");

    stud data_read[3];
    fread(&data_read, sizeof(data_read), 1, fptr);
    printf("%s %d %f\n", data_read[0].name,
data_read[0].id, data_read[0].average);
    printf("%s %d %f\n", data_read[1].name,
data_read[1].id, data_read[1].average);
    printf("%s %d %f\n", data_read[2].name,
data_read[2].id, data_read[2].average);

    fclose(fptr);
}
```

What is socket?

套接字

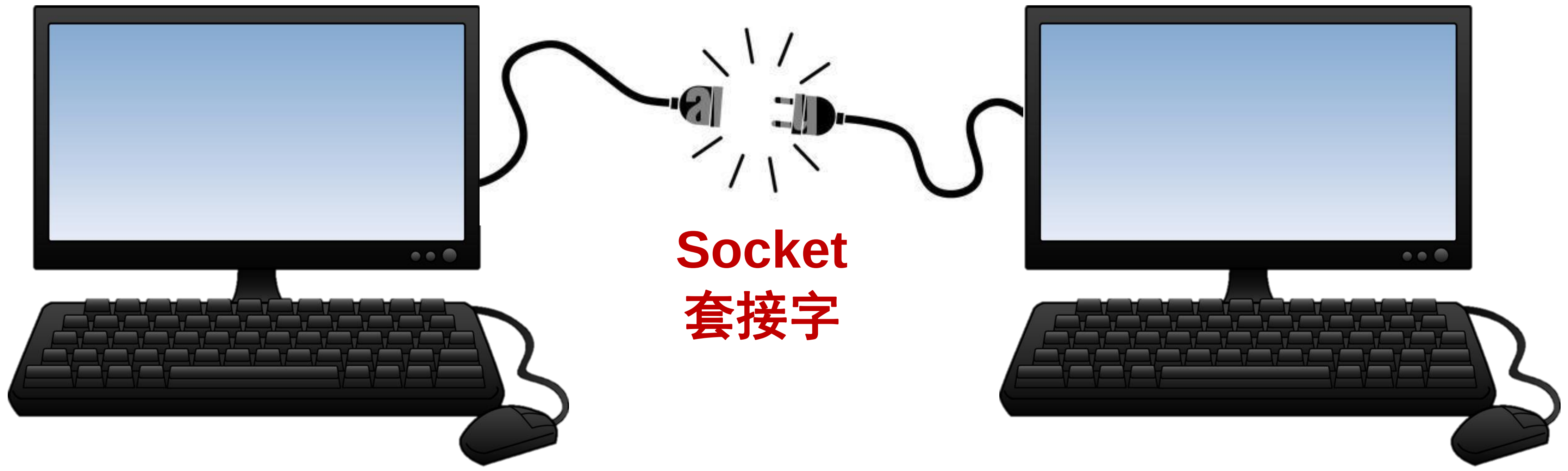
套接字（socket）是对**通信协议(TCP/UDP)**，**IP地址**，**端口**的函数封装，应用程序可以通过它建立网络连接，发送/接收网络传输的数据。



What is socket?

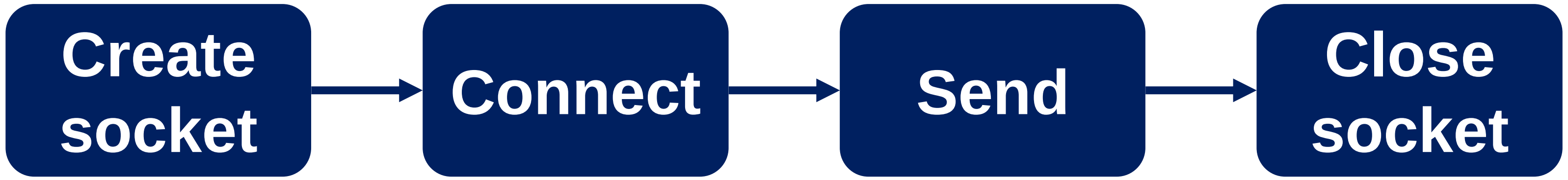
IP address	Port
10.0.0.201	1234

IP address	Port
10.0.0.101	101



What is socket?

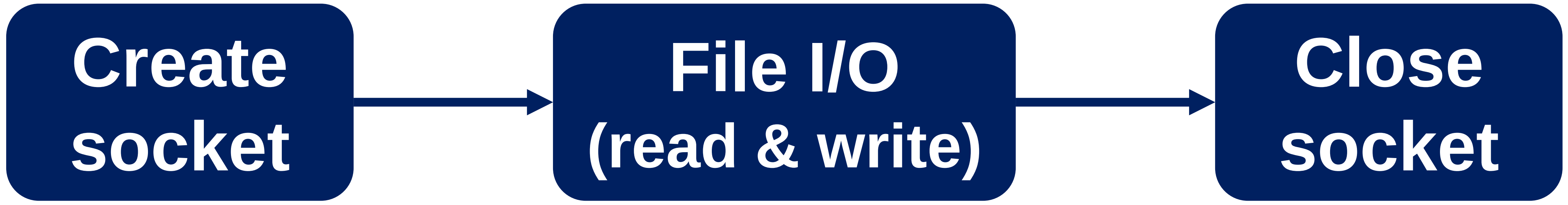
IP address	Port
10.0.0.201	1234



1. Socket socket = createSocket(**type = "TCP")**
2. connect(socket, **address = "1.2.3.4"**, **port = "80"**)
3. send(socket, "Hello, world!")
4. close(socket)

What is socket?

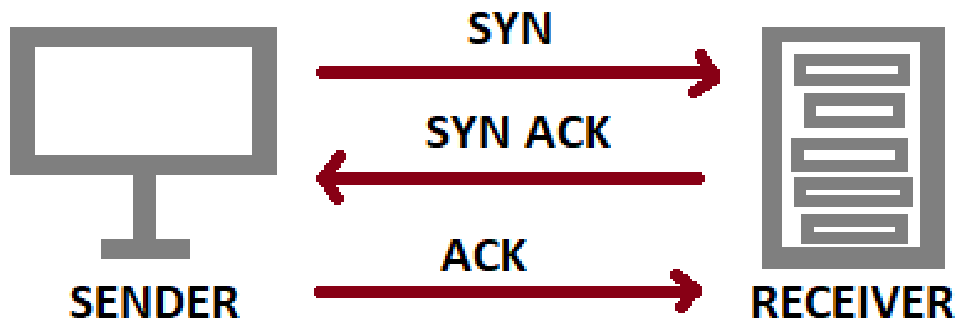
Use socket to create an internet connection, rest are file I/O!!!

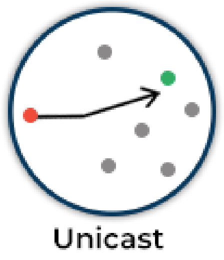


- Communication protocol
- IP address
- Porter

Communication protocol

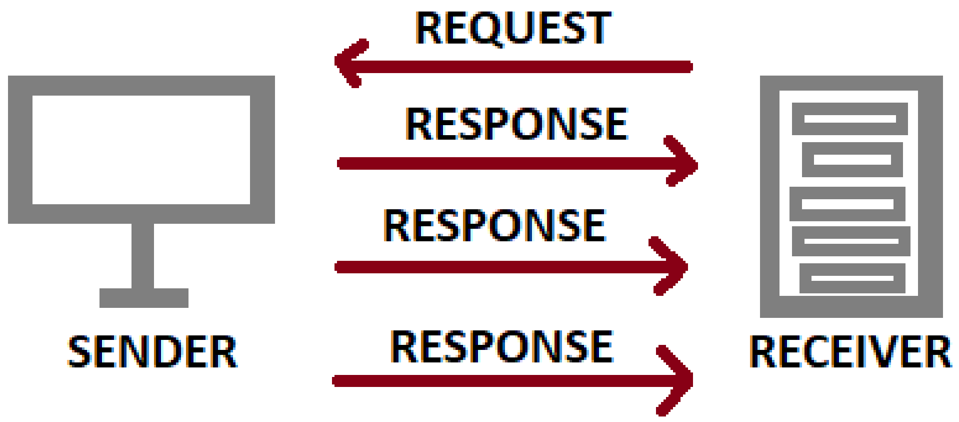
Transmission Control Protocol (TCP)

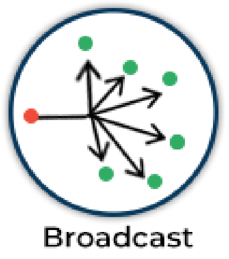




- Connection-oriented, 1-to-1
- Slower but reliable transfer
- Source intensive
- Typical applications: emails, web browsing, file transfer, etc.

User Datagram Protocol (UDP)



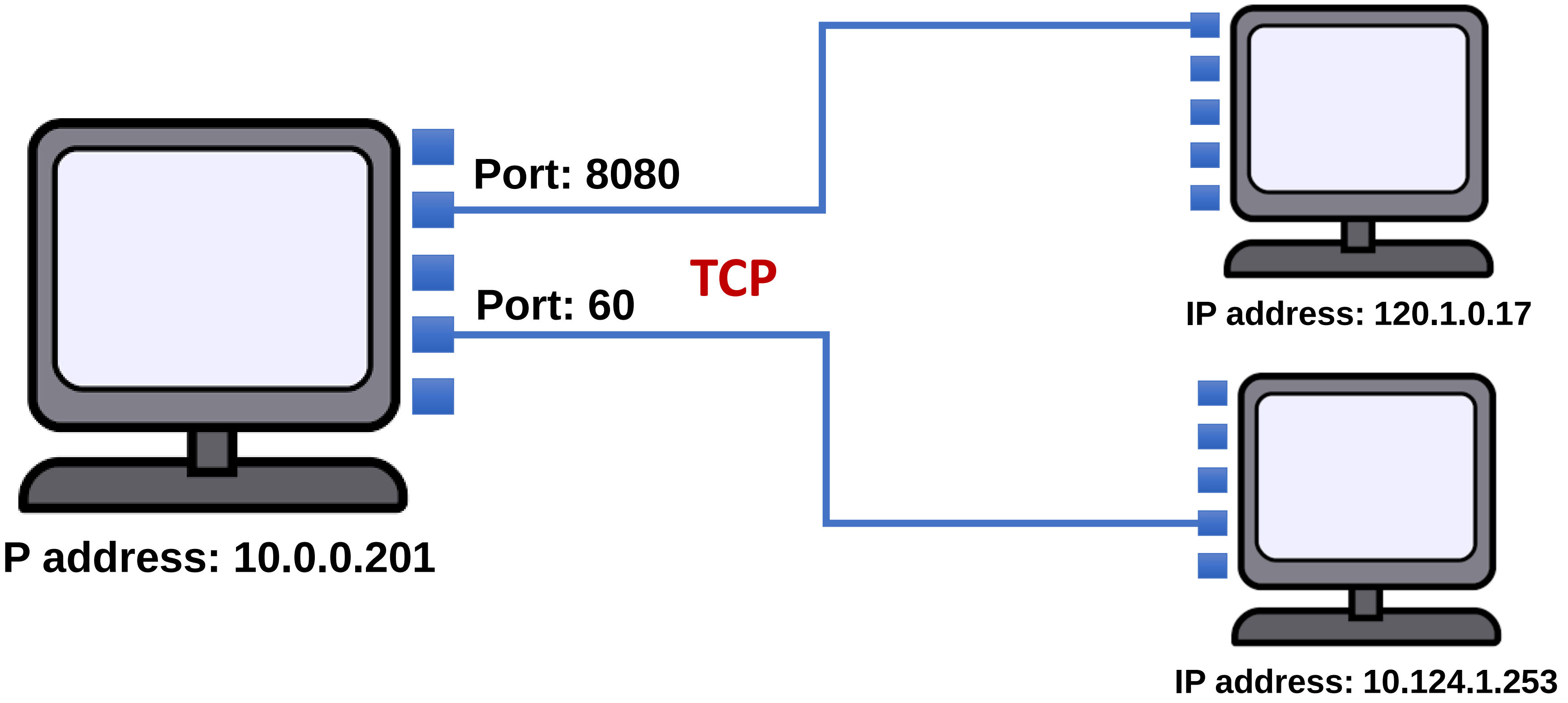


- Message-oriented, 1-to-N
- Faster but no guaranteed transfer
- Lightweight
- Typical applications: live streaming, online games, etc.

UDP isn't that bad in reality

IP address & port number

Socket address		IP address		Port number
10.0.0.201:8080	=	10.0.0.201	+	8080



IP address & port number

IP address	Port number
10.0.0.201	8080

It identifies a machine in the network, must be **unique**!

It identifies a particular **application or process** in a system.

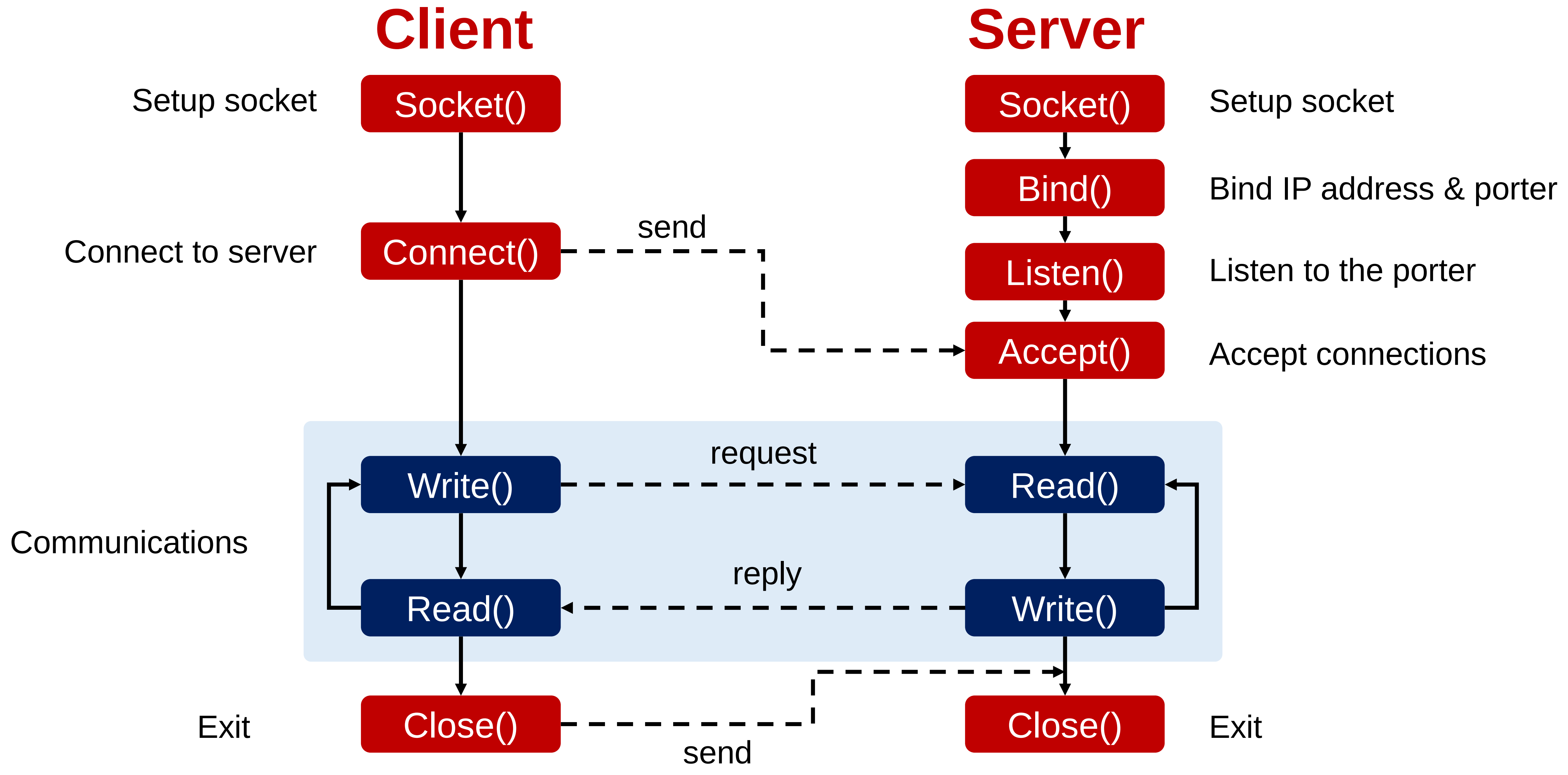
IPV4

192.168.5.18
4 octave x 8 bit = 32 bits（十进制）

IPV6

50b2:6400:0000:0000:6c3a:b17d:0000:10a9
8 octave x 16 bit = 128 bits（十六进制）

Overview of a socket

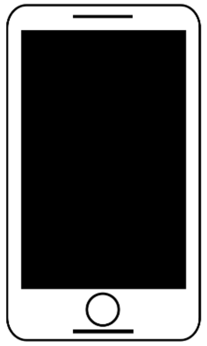
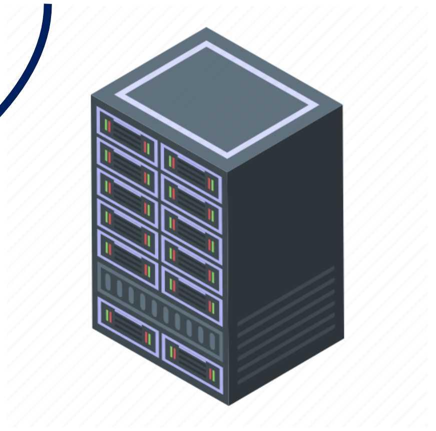


Case study: socket

Client.c

Connecting 2 nodes in a network to communicate with each other

Server.c

Case study: Server.c

```c
#define _WINSOCK_DEPRECATED_NO_WARNINGS
#include <winsock2.h>
#include <stdlib.h>
#include <conio.h>
#include <stdio.h>
#pragma comment(lib, "WS2_32")

void CInitSock_func(BYTE minorVer, BYTE majorVer)
{
    WSADATA wsaData;
    WORD sockVersion = MAKEWORD(minorVer, majorVer);
    if (WSAStartup(sockVersion, &wsaData) != 0)
    {
        exit(0);
    }
}
int main()
{
    CInitSock_func(2, 2);
```

根据版本号加载相应的库文件

选择socket的版本（副版本号，主版本号）

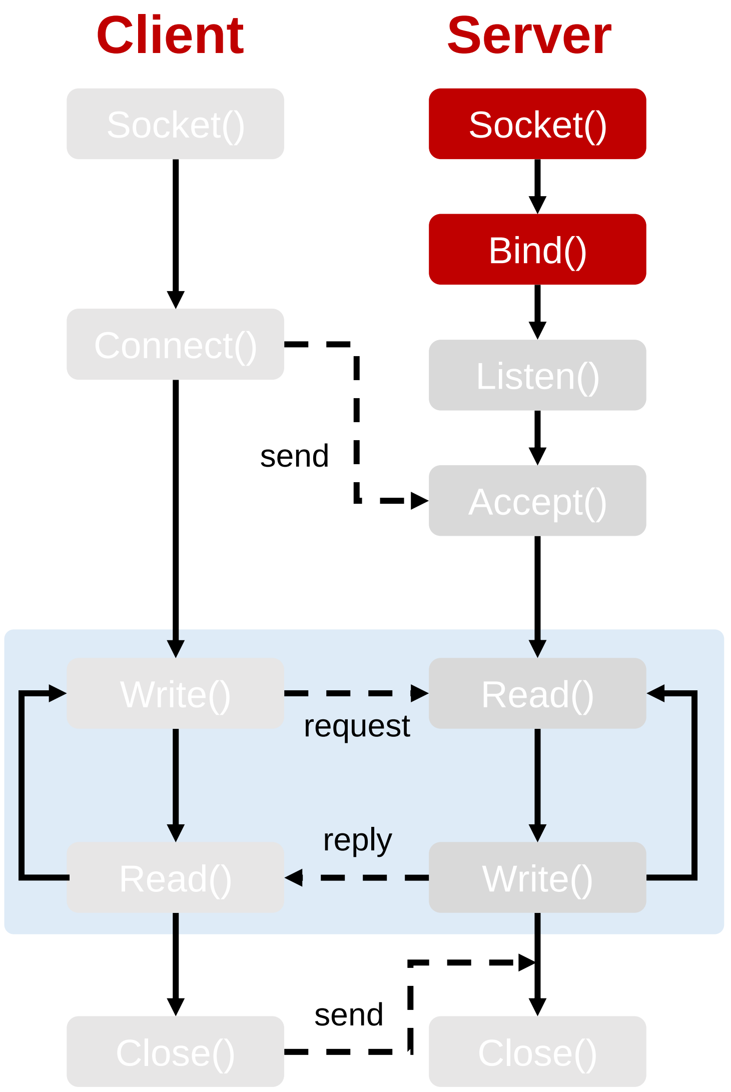
Case study: Server.c

```c
SOCKET sockfd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

if (sockfd == INVALID_SOCKET)
{
    printf("Failed socket() \n");
    return 0;
}

struct sockaddr_in sin;
sin.sin_family = AF_INET;
sin.sin_port = htons(4567);
sin.sin_addr.S_un.S_addr = INADDR_ANY; // IP和端口合并

int flag = bind(sockfd, (const struct sockaddr*)&sin,
sizeof(sin));

if (flag == SOCKET_ERROR)
{
    printf("Failed bind() \n");
    return 0;
}
```

创建一个套接字

定义IP address + port
用户可用端口：1024 ~ 49151

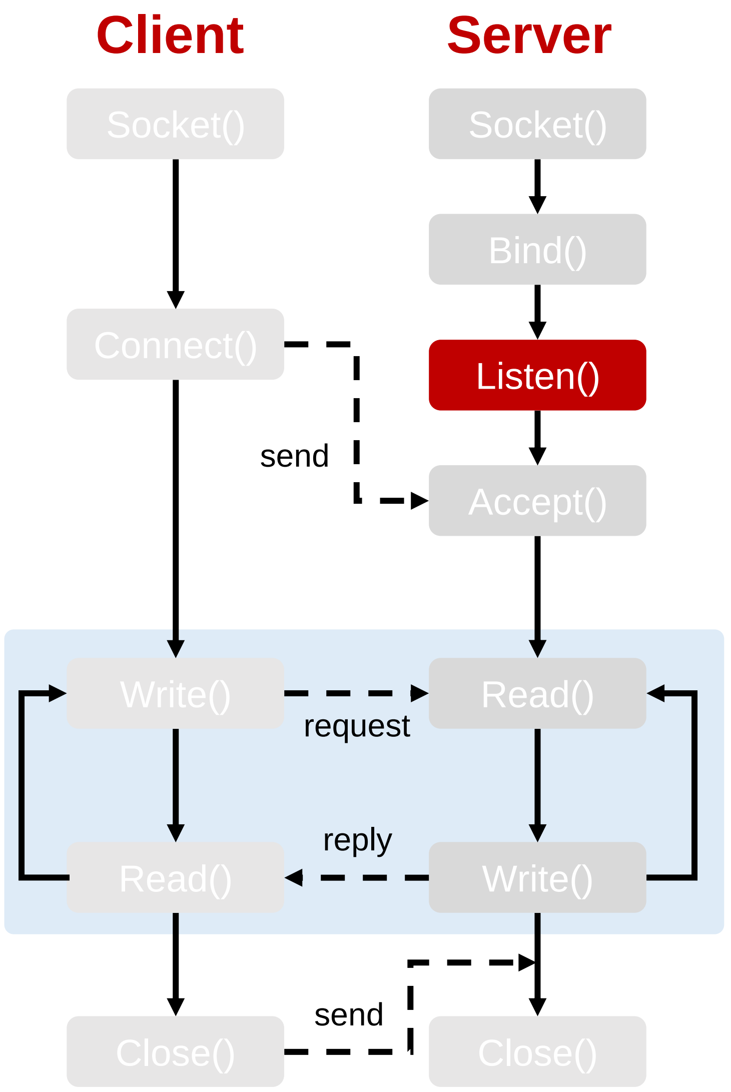
绑定socket和地址 (IP + port)



Case study: Server.c

进入监听模式
2 = 监听队列中允许保持的尚未处理的最大连接数

```
flag = listen(sockfd, 2);

if (flag == SOCKET_ERROR)
{
    printf("Failed listen() \n");
    return 0;
}
```

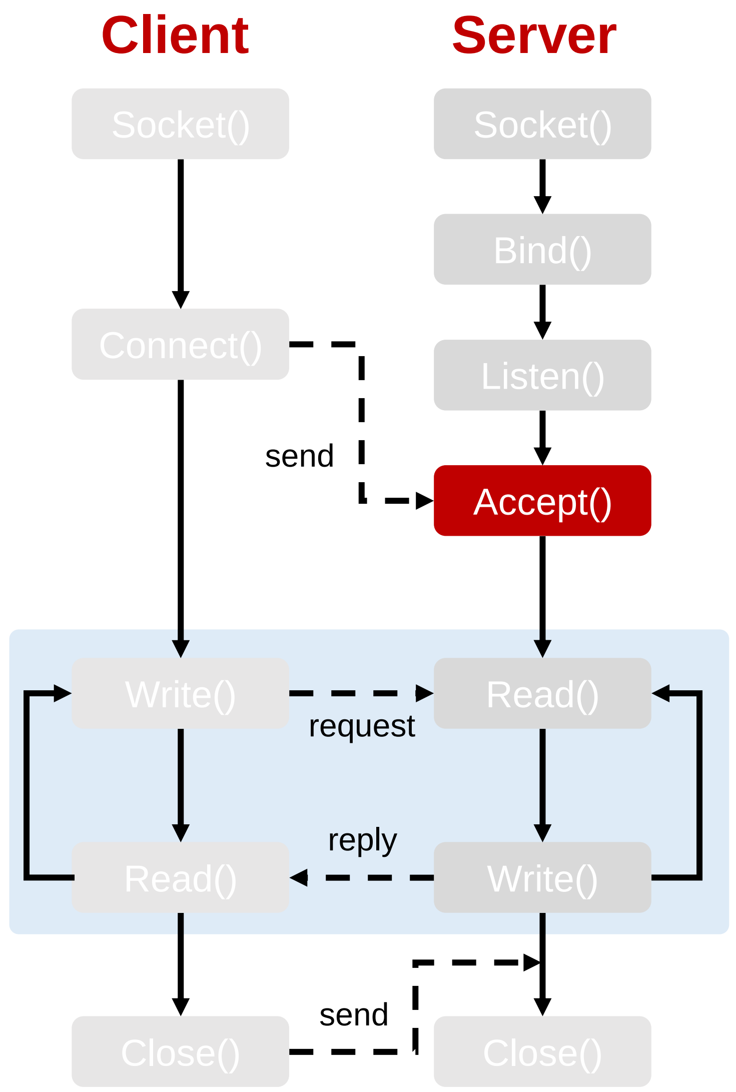


Case study: Server.c

接受用户的连接请求（阻塞）
阻塞：如果没有收到请求，程序会一直停留在这里等待

```
struct sockaddr_in remoteAddr;
int nAddrLen = sizeof(remoteAddr);


SOCKET sClient = accept(sockfd, &remoteAddr, &nAddrLen);
if (sClient == INVALID_SOCKET)
{
    printf("Failed accept()");
    return 0;
}
printf("接受到一个连接: %s \r\n", inet_ntoa(remoteAddr.sin_addr));
```

获得客户端IP地址



Case study: Server.c

```
while (1)
{
    char buff[256];
    int nRecv = recv(sClient, buff, 256, 0);
    if (nRecv > 0)
    {
        printf(" 接收到数据: %s\n", buff);
        send(sClient, buff, sizeof(buff), 0);
    }
}

closesocket(sClient);

closesocket(sockfd);
```

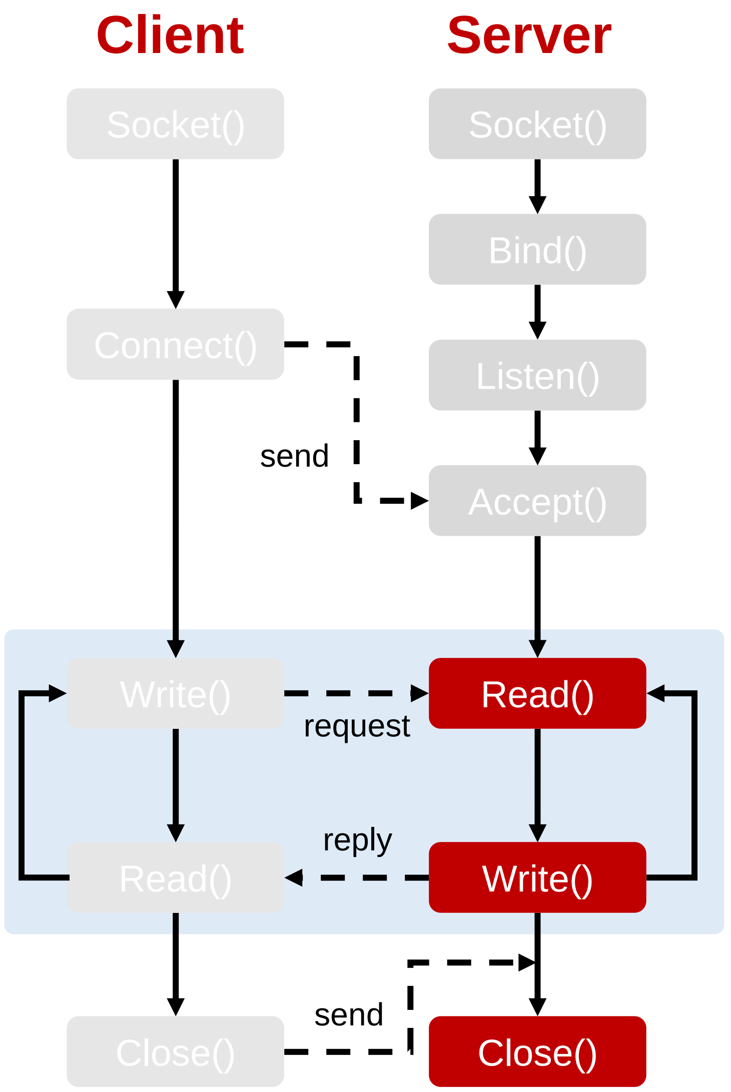
接收客户端的数据
阻塞：一直接收

长度

向客户端发送数据
阻塞：一直发送

关闭client socket

关闭server socket



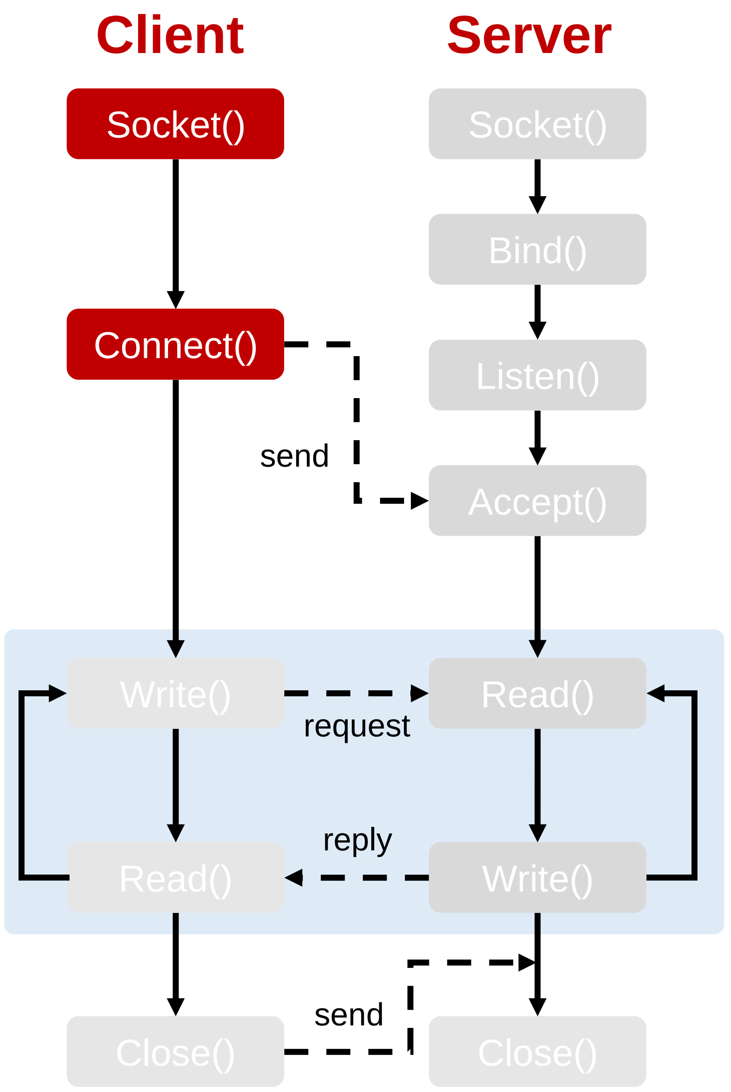
Case study: Client.c

```
SOCKET sockfd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

if (sockfd == INVALID_SOCKET)
{
    printf(" Failed socket() \n");
    return 0;
}

struct sockaddr_in servAddr;
servAddr.sin_family = AF_INET;
servAddr.sin_port = htons(4567);
servAddr.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");

if (connect(sockfd, (const struct sockaddr*)&servAddr, sizeof(servAddr))
== -1)
{
    printf(" Failed connect() \n");
    return 0;
}
```

创建套接字

设置服务器网址
127.0.0.1 为本地环回
自己发给自己

连接服务器



Case study: Client.c

```
SOCKET sockfd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

if (sockfd == INVALID_SOCKET)
{
    printf(" Failed socket() \n");
    return 0;
}

struct sockaddr_in servAddr;
servAddr.sin_family = AF_INET;
servAddr.sin_port = htons(4567);
servAddr.sin_addr.S_un.S_addr = inet_addr("192.168.3.75");

if (connect(sockfd, (const struct sockaddr*)&servAddr, sizeof(servAddr))
== -1)
{
    printf(" Failed connect() \n");
    return 0;
}
```

把IPV4地址填在这里
就可以连接这台电脑

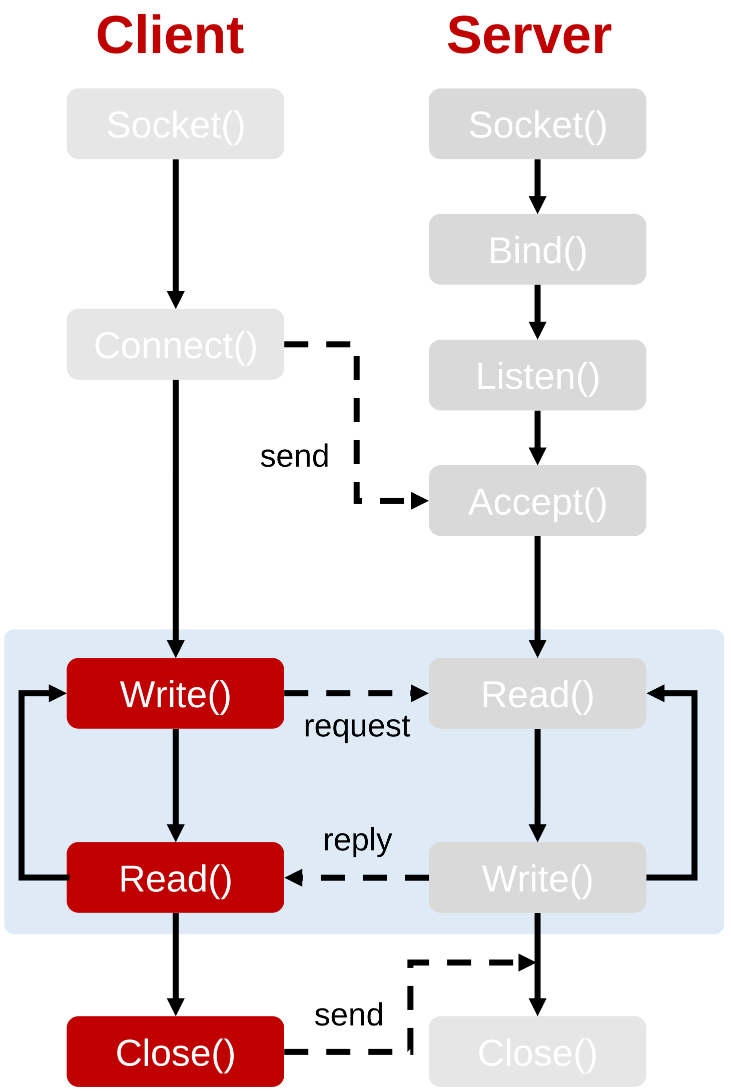
Case study: Client.c

```
char buff[256];
char szText[256];
while (TRUE)
{
     int length = sizeof(servAddr);
     int nRecv = recvfrom(sockfd, buff, 256, 0, (struct sockaddr*)&servAddr,
     &length);
     if (nRecv > 0)
     {
          printf("接收到数据: %s\n", buff);
     }
     gets_s(szText, 256); // user input
     sendto(sockfd, szText, sizeof(szText), 0, (const struct sockaddr*)&servAddr,
     sizeof(servAddr));
}

closesocket(sockfd);
```

从服务器接收数据

向服务器发送数据

关闭套接字



5 questions

1. Which of following statement is correct for opening a binary file? ( )

A. FILE *f = fwrite("test.bin", "b");
B. FILE *f = fopenb("test.bin", "w");
C. FILE *f = fopen("test.bin", "wb");
D. FILE *f = fwriteb("test.bin");

2. What is the function of following code? ( )

A. Copy file
B. Calculate the number of characters
C. Calculate the number of words
D. Calculate the number of rows

```
int main() {

    FILE* fp = fopen("fname.dat", "r"));

    int num = 0;
    while (fgetc(fp) != EOF)
    {
        num++;
    }
    printf("num=%d\n", num);

    fclose(fp);

    return 0;
}
```

5 questions

3. Which of following file format can be used to store a struct? ( )
A. dat B. txt C. bin D. csv

4. What does a client need when building a socket connection with server? ( )
A. Server’s IP address and port number
B. Server’s physical location
C. Server’s IP address only, port number will be assigned automatically
D. Server’s IP address and the format of transmitted data

5. TCP is a data secured transmission protocol, while UDP is more focused on the transmission speed and low cost. Yes | No

Content

1. Pointer
2. Self-defined types
3. I/O
4. **Head files**

What is head file?

A header file is **a file with extension .h,** which contains C function declarations and macro definitions that can be used by different source files.

This is c file $\longrightarrow$ run.c

This is head file $\longrightarrow$ stdio.h

What is head file?

Two types of head file

C compiler defined head file

stdio.h

User defined head file

myFun.h

Avoid using names of C compiler defined head files!

Why using head file?

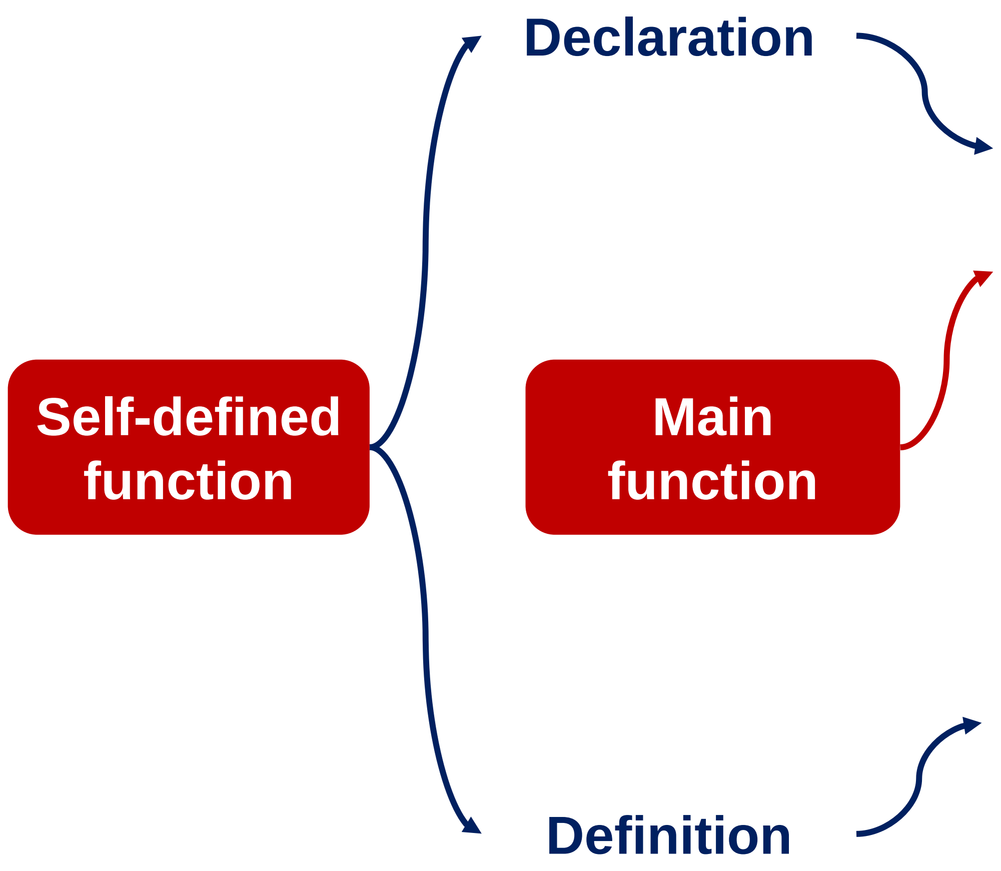
When your program grows large, it’s impossible to keep all functions in one file.

You can move parts of a program (functions) to separate files, and link them by head file.

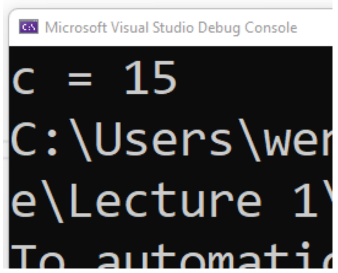
You already used C
Compile defined head file

```
#include<stdio.h>

main()
{
    printf("Hello World!");
}
```

An example of function



```c
#include<stdio.h>

int sum(int x, int y);

main()
{
    int a = 10, b = 5;
    int c = sum(a, b);
    printf("c = %d", c);
}

int sum(int x, int y)
{
    return x + y;
}
```



```
c = 15
C:\Users\wer
e\Lecture 1\
To automatic
```

How to use head files?

方程声明放入头文件 $\longrightarrow$ myFun.h

方程定义放入c文件 $\longrightarrow$ myFun.c

How to use head files?

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

```
int sum(int x, int y);
```

```
main()
{
    int a = 10, b = 5;
    int c = sum(a, b);
    printf("c = %d", c);
}
```

```
int sum(int x, int y)
{
    return x + y;
}
```

Declaration

Main

Definition

myFun.h
(declaration)

```
int sum(int x, int y);
```

run.c

```
#include<stdio.h>
#include"myFun.h"

main()
{
    int a = 10, b = 5;
    int c = sum(a, b);
    printf("c = %d", c);
}
```

myFun.c
(definition)

```
#include"myFun.h"

int sum(int x, int y)
{
    return x + y;
}
```

How to use head files?

Include the declaration of sum() by including myfun.h

这可以看成是函数库

```c
#include<stdio.h>
#include"myFun.h"

main()
{
    int a = 10, b = 5;
    int c = sum(a, b);
    printf("c = %d", c);
}
```

run.c

```c
int sum(int x, int y);
```

myFun.h

(declaration)

```c
#include"myFun.h"

int sum(int x, int y)
{
    return x + y;
}
```

myFun.c

(definition)

How to use head files?

Use **< >** to include C compiled head file

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

Use **" "** to include user defined head file

```
#include"myFun.h"
```

Step-by-step to use head files

① write head file **xxx.h** (declare functions)

```c
int sum(int x, int y);
```

myFun.h

② write c file **xxx.c** (include xxx.h, define functions)

```c
#include"myFun.h"

int sum(int x, int y)
{
    return x + y;
}
```

myFun.c

③ write **run.c** (include xxx.h, call functions)

```c
#include<stdio.h>
#include"myFun.h"

main()
{
    int a = 10, b = 5;
    int c = sum(a, b);
}
```

run.c

You can use head files of C

```
#include<stdio.h>

#include<iostream>

#include<math.h>

#include<string.h>

…

#include<pthread.h>  线程
```

5 questions

1. Head file can be used to declare functions. Yes | No

2. Which keyword can be used to include a head file? ( )
A. #include B. #define C. typedef D. extern

3. What is the difference between #include “XXX.h” and #include <XXX.h>?

4. A head file cannot include another head file. Yes | No

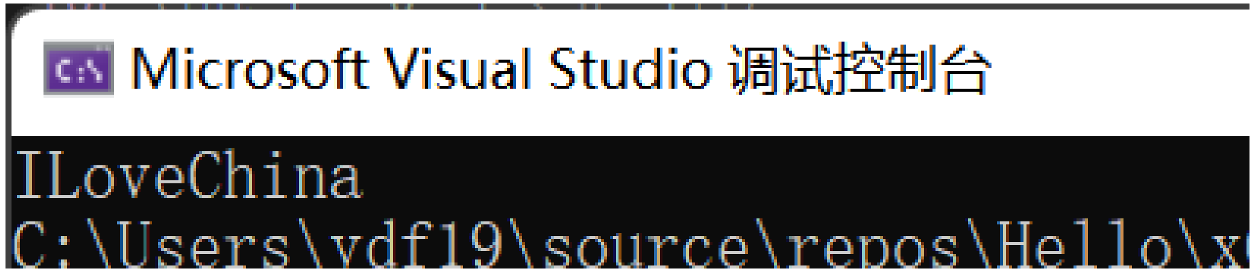
5. The name of head file and the file to define the functions in the head file must be the same, for example, myfun.h must have myfun.c for its implementation. Yes | No

Assignment

1. Write a function to implement “string insertion” which means to insert a string str2 at a specific position of the string str1

a) Since the length of a static string aka a static array of chars can not be changed, you can use “malloc()” or “calloc()” to create a new string and return it
b) You can define the function like this: `char* insert(char* str1, char* str2, int n)`
c) Test input str1 = “IChina”, str2 = “Love”

```c
void main()
{
    char s1[] = "IChina";
    char s2[] = "Love";
    char* ss = insert(s1, s2, 1);
    printf("%s", ss);
}
```


Microsoft Visual Studio 调试控制台

```
ILoveChina
C:\Users\vdf19\source\repos\Hello\x
```

Assignment

2. The class is choosing the class president, there are two candidates, each student can make one vote, the voting results are stored in a bin file. Write a program to read the bin file and print the voting result (print who is the class president).

a) The struct that contains the voting results is shown below
b) The bin file will be uploaded to bb
c) 10 students participated in the vote

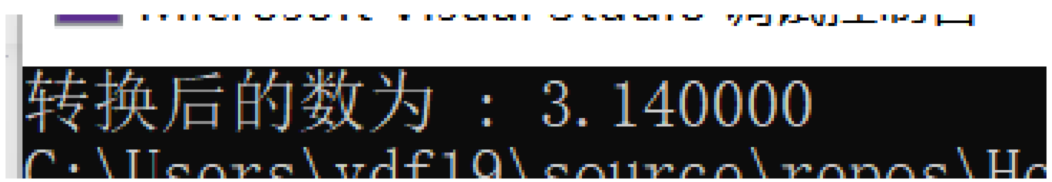
```
typedef enum vote {egon, erick}Vote;

typedef struct stu
{
int stu_number;
Vote result;
}Stu;

Stu students[10]
```

Assignment

3. CSV file are commonly used in the field of machine learning, write a program to implement reading csv file. You will be given a csv file which contains 10 float numbers. Read the file and print the sum of the 10 numbers.

a) A CSV file is a text file separated by “,”
b) You can use strtok / strtok_s() to replace all the “,” with “\0”
c) There is an example of the using of strtok() / strtok_s() on the next page
d) The float numbers is stored as strings in the csv file, you can use atof() which is defined in <math.h> to convert strings(ASCII) to float numbers. Here is an exmaple

```c
#include <stdio.h>
#include <math.h>

int main()
{
char float_in_ACCII[] = "3.14";
float num_after_convert = atof(float_in_ACCII);
printf("转换后的数为 : %f", num_after_convert);
}
```



```
转换后的数为 : 3.140000
C:\Users\ydf19\source\repos\H
```

Assignment

You can use strtok in most C IDE

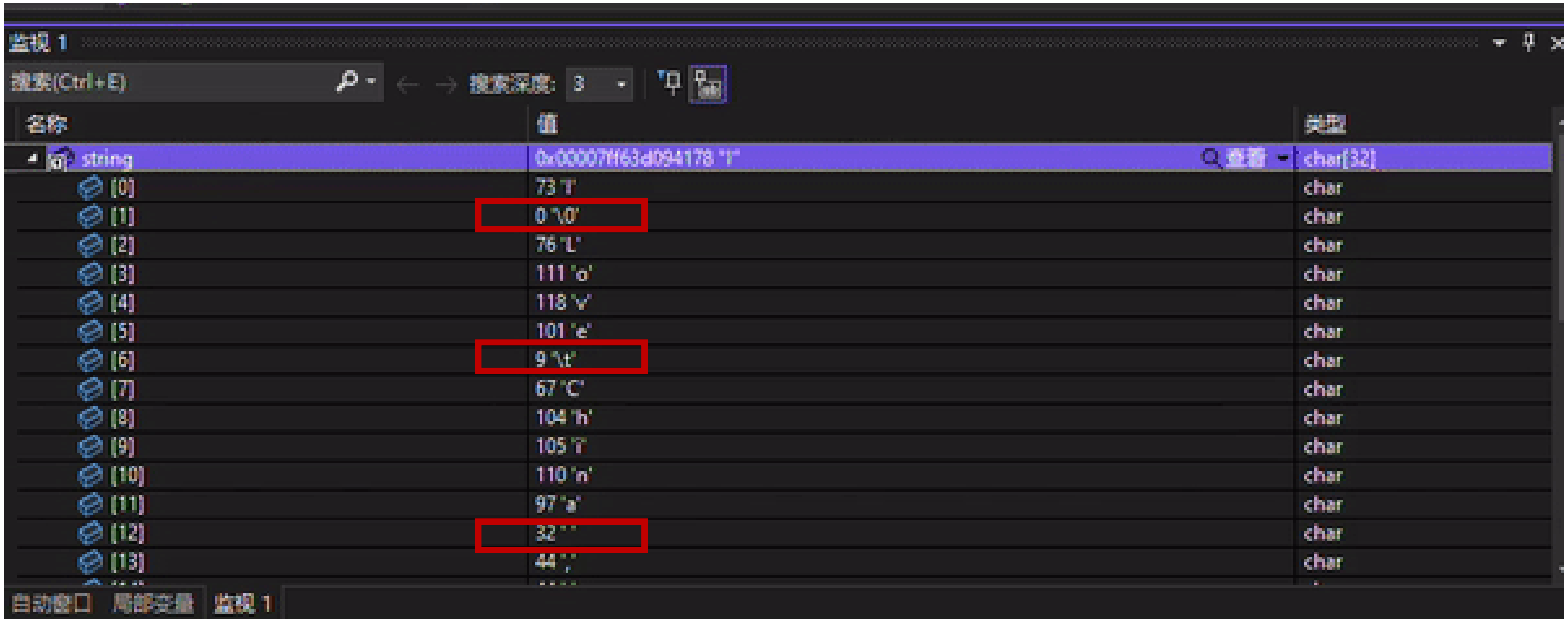
```
#include <stdio.h>
#include <string.h>

char string[] = "I Love\tChina ,,I\nLove SUS  Tech";
char seps[] = " ,\t\n";
char* token;
int main(void)
{
printf("Tokens:\n");
token = strtok(string, seps);//将指针指向I
while (token != NULL)
{
printf("%s\n", token);
token = strtok(NULL, seps);//去掉标点符号，将指针指向下一个字符串(Love, China ……)
}
return 0;
}
```

You must use strtok_s in VS 2022

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

char string[] = "I Love\tChina ,,I\nLove SUS  Tech";
char seps[] = " ,\t\n";
char* token;
int main(void)
{
char* flag = NULL;//这个是strtok_s要求的，用于防止多线程造成的错误
printf("Tokens:\n");
token = strtok_s(string, seps, &flag);
while (token != NULL)
{
printf("%s\n", token);
token = strtok_s(NULL, seps, &flag);
}return 0;
}
```

Assignment

Strtok() can replace “,\n\t” with “ ” and move the pointer which is returned by strtok() to the next position



监视 1

搜索(Ctrl+E) 搜索深度: 3

名称	值	类型
string	0x00007ff63d094178 "I"	char[32]
[0]	73 'I'	char
[1]	0 '\0'	char
[2]	76 'L'	char
[3]	111 'o'	char
[4]	118 'v'	char
[5]	101 'e'	char
[6]	9 '\t'	char
[7]	67 'C'	char
[8]	104 'h'	char
[9]	105 'i'	char
[10]	110 'n'	char
[11]	97 'a'	char
[12]	32 ' '	char
[13]	44 ','	char

自动窗口 局部变量 监视 1