Week 4 - Lecture 3
Pointers
**Edited by: Heshan Du**

Autumn 2024

Overview

- Declaration and initialisation
- Pointer to Constant vs. const Pointer
- Pointers and arrays
  - String literals
- **Array of pointers**
- **Pointer arithmetic (e.g., subtracting, comparing)**

Arrays of Pointers

- Every element in the array is a pointer to the same data type

- char *arr[3]; array of 3 pointers to arrays of characters

  - Common use i.e., array of strings

char str[3][40];
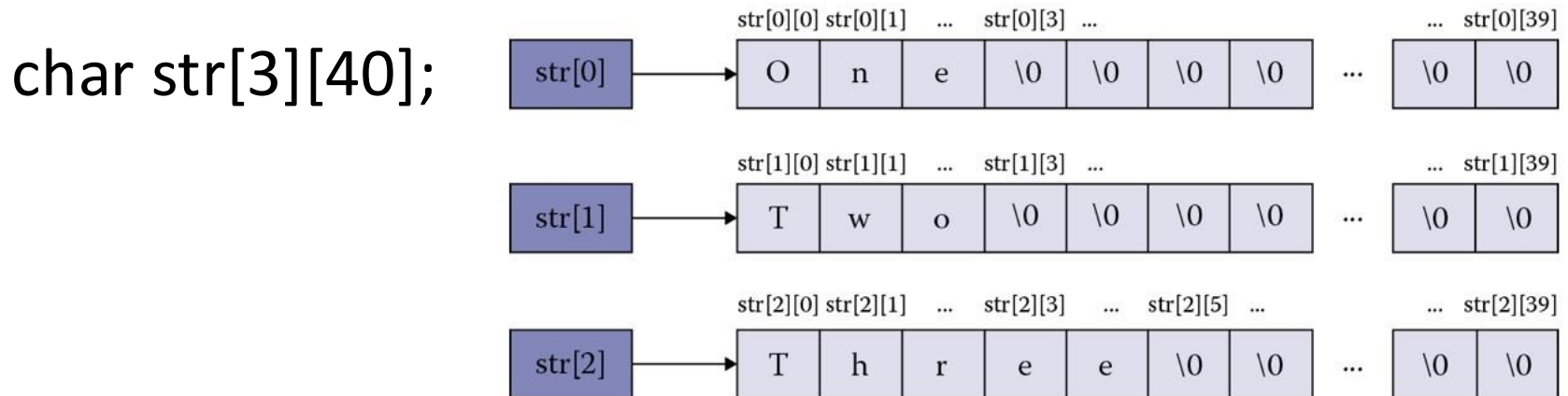
Arrays of Pointers (2)

- A common use of an array of pointers is to form an array of strings, referred to simply as a string array.

- Consider the definition of string array **suit**, which might be useful in representing a deck of cards.

- const char ***suit**[4] = { "Hearts", "Diamonds", "Clubs", "Spades" };
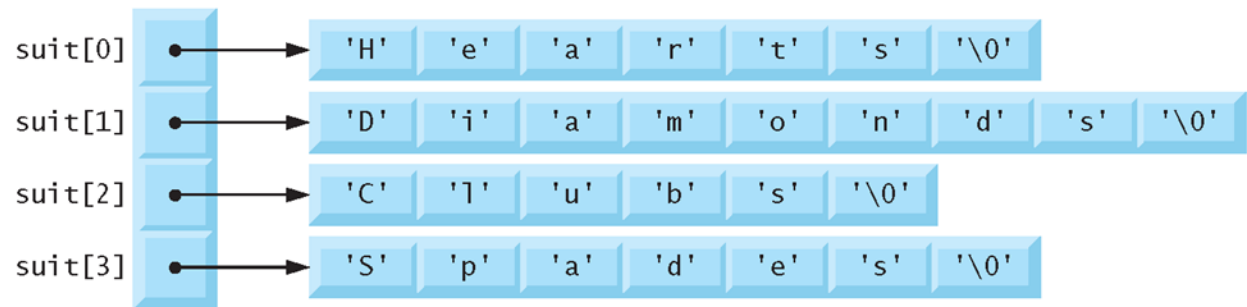
Arrays of Pointers (3)

- The **suits** could have been placed in a two-dimensional array.
    - Such a data structure would have to have a fixed number of columns per row, and that number would have to be as large as the largest string.

    - Therefore, considerable memory could be wasted when storing a large number of strings of which most were shorter than the longest string.
- **Because of this, we use arrays of pointers!**

Q1: What will be shown here?

- int *arr[3], i, p[3] = {10, 20, 30};

```
for(i = 0; i < 3; i++){
    arr[i] = &p[i];
    printf("%d", *arr[i]);
}
```

arr

p

0

0 **10**

1

1 20

2

2 30
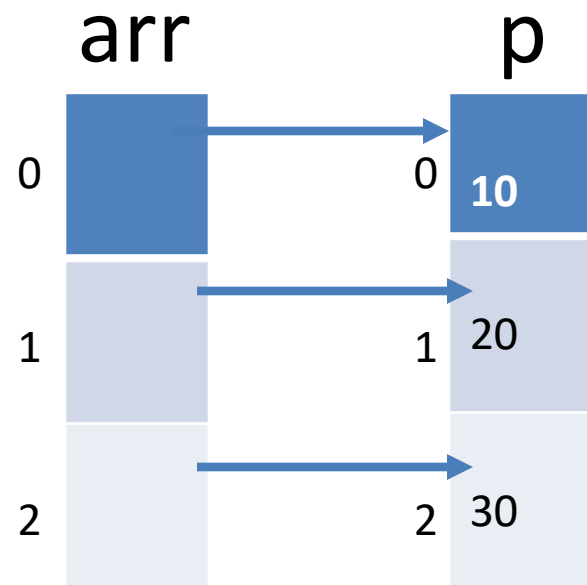
Q2: What are first chars?

- char *arr[3];
  int i;
  arr[0] = "This is";
  arr[1] = "a new";
  arr[2] = "message";
  for(i = 0; i < 3; i++)
       printf("Text: %s\tFirst char: %c\n", arr[i], *arr[i]);

Overview

- Declaration and initialisation
- Pointer to Constant vs. const Pointer
- Pointers and arrays
  - String literals
- Array of pointers
- **Pointer arithmetic (e.g., subtracting, comparing)**

Pointer Arithmetic

$$ptr = ptr + n;$$

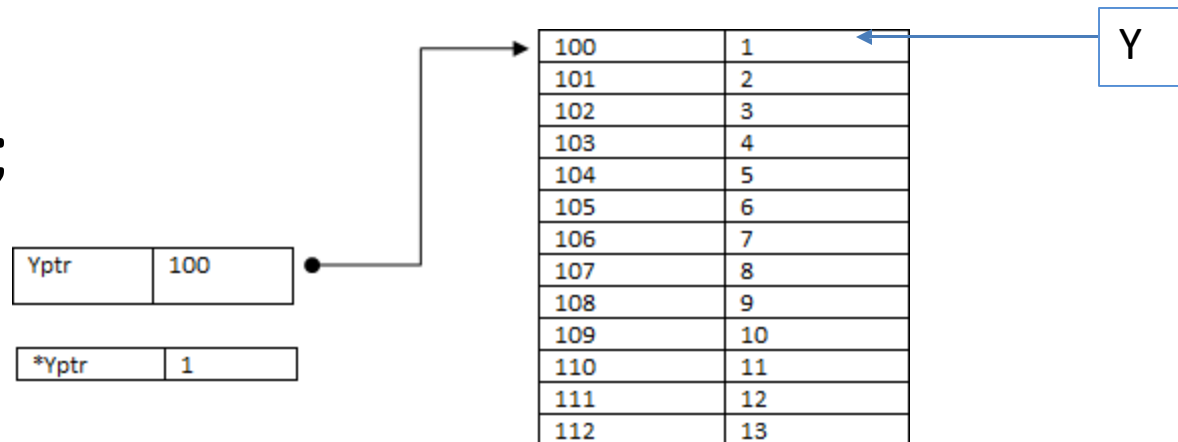- **char: ptr** is increased by n; char size is 1 byte.
- **int or** float: ptr is increased by n * 4, int and float size is 4 bytes.
- **double: ptr** is increased by n * 8; double size is 8 bytes.

Remember this?!

- A variable name *directly* references a value, a pointer ***indirectly*** references a value.

int Y = 1;
int *Yptr;
Yptr = &Y;

Source: http://www.exforsys.com/tutorials/c-language/c-pointers.html

Pointer Arithmetic: Example

- int *ptr, i;
  ptr = &i;
  printf("Address = %p\n", ptr);
  ptr++;
  printf("Address = %p\n", ptr);

The second address will be 4 bytes higher than the first one

University of Nottingham
UK | CHINA | MALAYSIA

Subtracting Pointers

- Only if both pointers refer to the **same data type**, the result of subtracting pointers indicates **the number of data items between them**

- Suppose ptr1 and ptr2 point to two integer variables stored in addresses 1000 and 1040 respectively

- **(ptr2 - ptr1) != 40**

- **(ptr2 - ptr1) == (40 / 4) == 10**

Comparing Pointers

- Only if both point to members of the **same data structure**

- Operators: ==, !=, >, <, >= and <=

- To check if two pointers point to **the same address**
  - if(ptr1 == ptr2) or if(ptr1 != ptr2)

Q3: explain how this Pointer works?

```c
int *ptr, i;
ptr = &i;
printf("Address = %p\n", ptr);
ptr -= 10;
printf("Address = %p\n", ptr);
```

Q4: What is the value of i, j and k?

```
int *ptr, i = 10, j = 20, k = 30;
ptr = &i;
*ptr = 40;
ptr = &j;
*ptr += i;
ptr = &k;
*ptr += i + j ;
printf("i = %d j = %d k = %d\n", i, j, k);
```

i

10

1000

ptr

500

j

20

2000

k

30

3000

Q4: What is the value of i, j and k?

```
int *ptr, i = 10, j = 20, k = 30;
ptr = &i;
*ptr = 40;
ptr = &j;
*ptr += i;
ptr = &k;
*ptr += i + j ;
printf("i = %d j = %d k = %d\n", i, j, k);
```
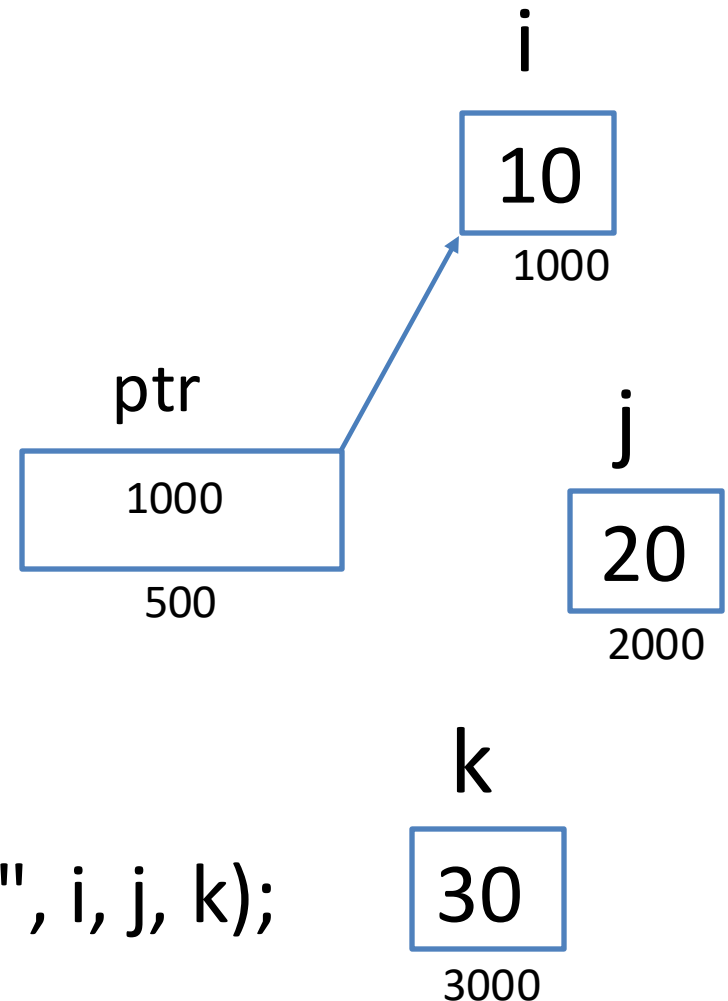
i

10

1000

ptr

1000

500

j

20

2000

k

30

3000

Q4: What is the value of i, j and k?

int *ptr, i = 10, j = 20, k = 30;

ptr = &i;

*ptr = 40;

ptr = &j;

*ptr += i;

ptr = &k;

*ptr += i + j ;

printf("i = %d j = %d k = %d\n", i, j, k);

i

40

1000

ptr

1000

500

j

20

2000

k

30

3000

Q4: What is the value of i, j and k?

```
int *ptr, i = 10, j = 20, k = 30;
ptr = &i;
*ptr = 40;
ptr = &j;
*ptr += i;
ptr = &k;
*ptr += i + j ;
printf("i = %d j = %d k = %d\n", i, j, k);
```

i

40

1000

ptr

2000

500

j

20

2000

k

30

3000

Q4: What is the value of i, j and k?

int *ptr, i = 10, j = 20, k = 30;

ptr = &i;
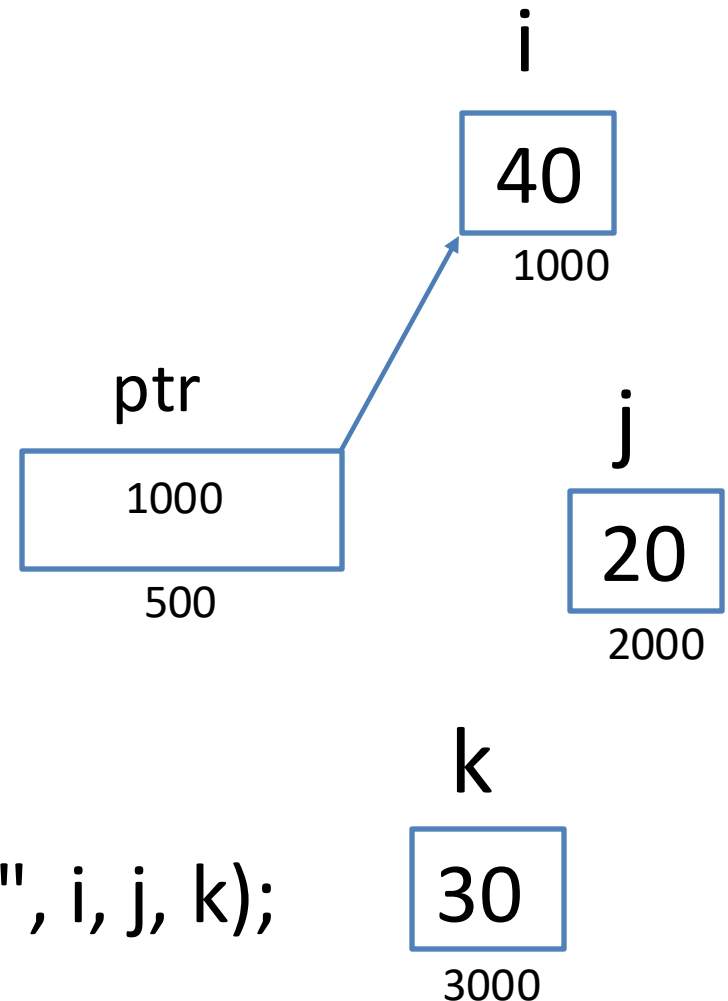
*ptr = 40;

ptr = &j;   ***ptr = *ptr + i;**

==*ptr += i;==   **=   20 + 40**

ptr = &k;

*ptr += i + j ;

printf("i = %d j = %d k = %d\n", i, j, k);
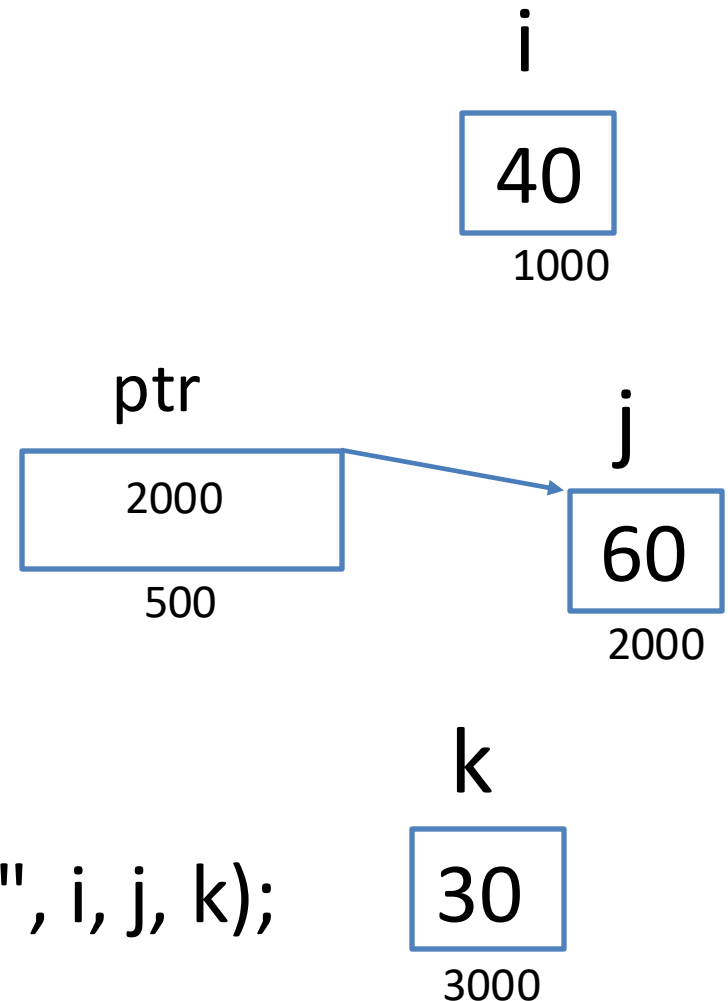
i

40

1000

ptr

2000

500

j

60

2000

k

30

3000

University of
Nottingham
UK | CHINA | MALAYSIA

Q4: What is the value of i, j and k?

```
int *ptr, i = 10, j = 20, k = 30;
ptr = &i;
*ptr = 40;
ptr = &j;
*ptr += i;
ptr = &k;
*ptr += i + j ;
printf("i = %d j = %d k = %d\n", i, j, k);
```

i

40

1000

ptr

3000

500

j

60

2000

k

30

3000

Q4: What is the value of i, j and k?

int *ptr, i = 10, j = 20, k = 30;

ptr = &i;

*ptr = 40;

ptr = &j;
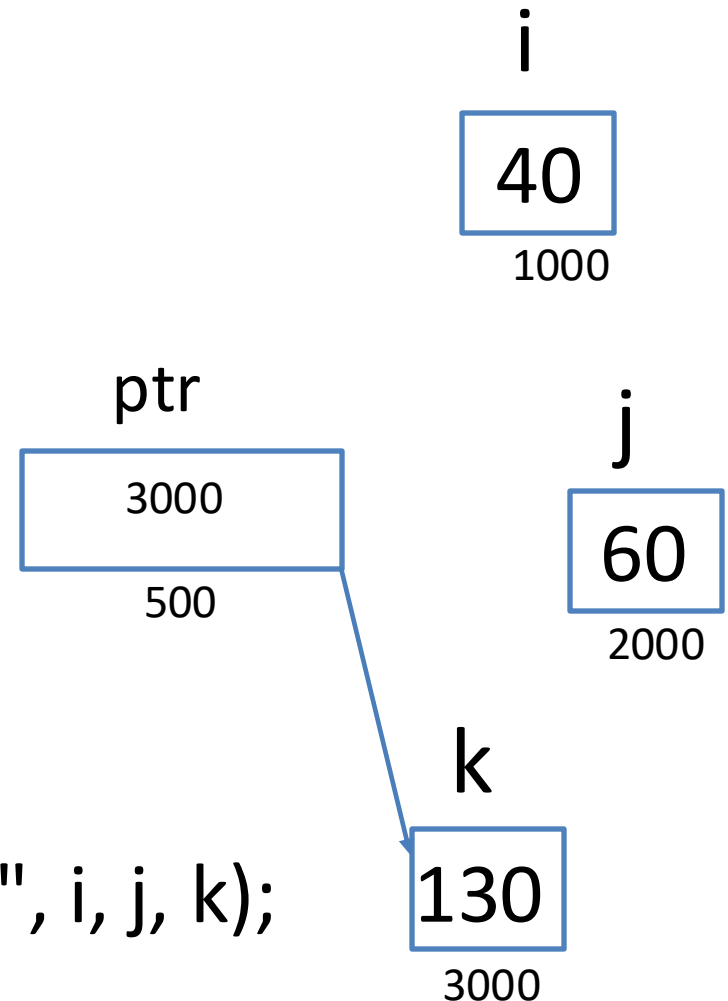
*ptr += i;

ptr = &k;
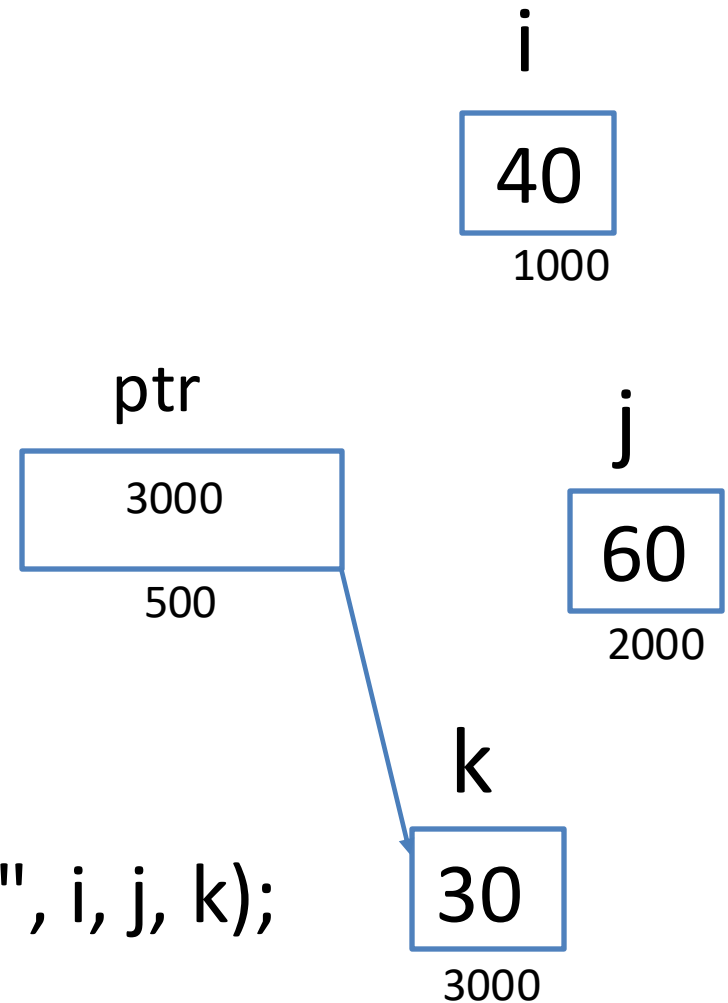
*ptr += i + j ;

printf("i = %d j = %d k = %d\n", i, j, k);

*ptr = *ptr + i + j;
= 30 + 40 + 60

i

40

1000

ptr

3000

500

j

60

2000

k

130

3000

Q5: What is the value of j?

- int *ptr1, *ptr2, i = 10, j = 20;
  ptr1 = &i;
  *ptr1 = 150;
  ptr2 = &j;
  *ptr2 = 50;
  ptr2 = ptr1;
  *ptr2 = 250;
  ptr2 = &j;
  *ptr2 += *ptr1;
  printf("Val = %d\n", j);

ptr1

500

ptr2

600

i

10

1000

j

20

2000
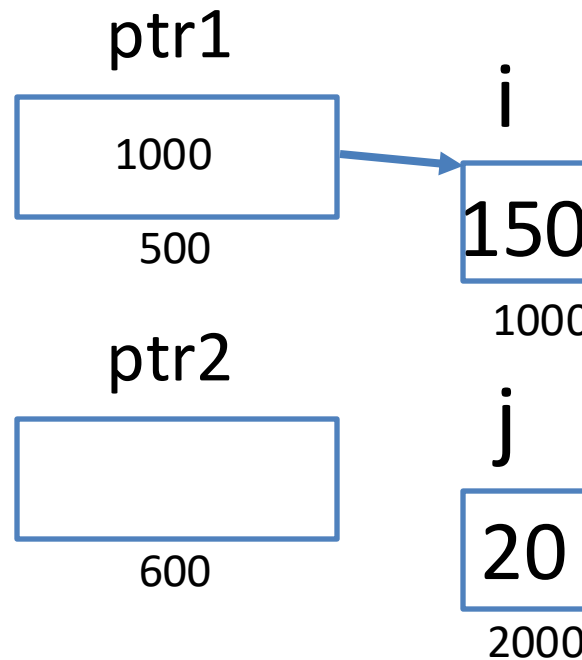
Q5: What is the value of j?

- int *ptr1, *ptr2, i = 10, j = 20;
  ptr1 = &i;
  *ptr1 = 150;
  ptr2 = &j;
  *ptr2 = 50;
  ptr2 = ptr1;
  *ptr2 = 250;
  ptr2 = &j;
  *ptr2 += *ptr1;
  printf("Val = %d\n", j);

ptr1

1000

500

i

150

1000

ptr2

600

j

20

2000

Q5: What is the value of j?

- int *ptr1, *ptr2, i = 10, j = 20;
ptr1 = &i;
*ptr1 = 150;
ptr2 = &j;
*ptr2 = 50;
ptr2 = ptr1;
*ptr2 = 250;
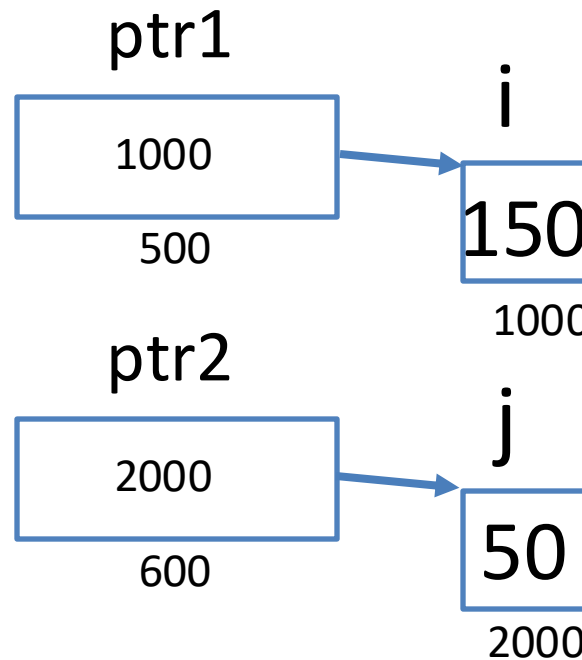ptr2 = &j;
*ptr2 += *ptr1;
printf("Val = %d\n", j);

ptr1

1000

500

i

150

1000

ptr2

2000

600

j

50

2000

Q5: What is the value of j?
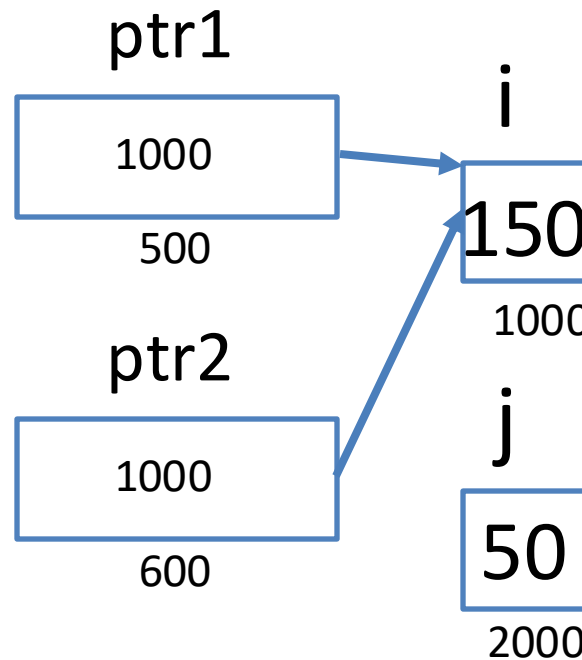
- int *ptr1, *ptr2, i = 10, j = 20;
  ptr1 = &i;
  *ptr1 = 150;
  ptr2 = &j;
  *ptr2 = 50;
  ptr2 = ptr1;
  *ptr2 = 250;
  ptr2 = &j;
  *ptr2 += *ptr1;
  printf("Val = %d\n", j);

ptr1

1000
500

i

150
1000

ptr2

1000
600

j

50
2000

Q5: What is the value of j?

- int *ptr1, *ptr2, i = 10, j = 20;
  ptr1 = &i;
  *ptr1 = 150;
  ptr2 = &j;
  *ptr2 = 50;
  ptr2 = ptr1;
  *ptr2 = 250;
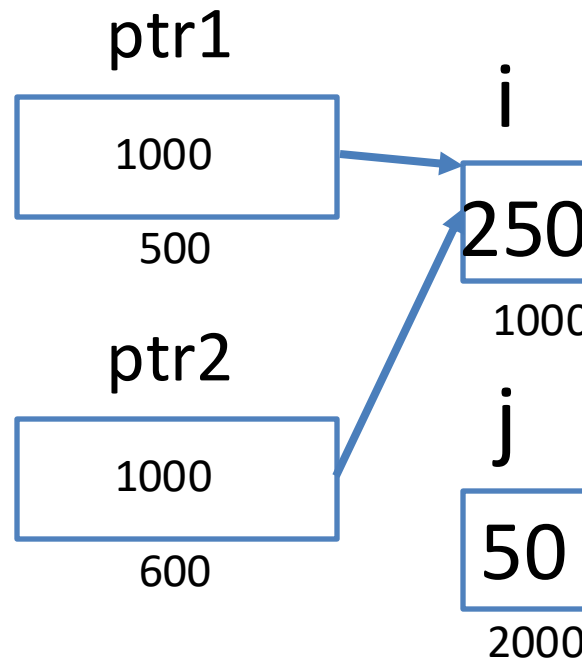 ptr2 = &j;
  *ptr2 += *ptr1;
  printf("Val = %d\n", j);

ptr1

1000

500

i

250

1000

ptr2

1000

600

j

50

2000

Q5: What is the value of j?

- int *ptr1, *ptr2, i = 10, j = 20;
  ptr1 = &i;
  *ptr1 = 150;
  ptr2 = &j;
  *ptr2 = 50;
  ptr2 = ptr1;
  *ptr2 = 250;
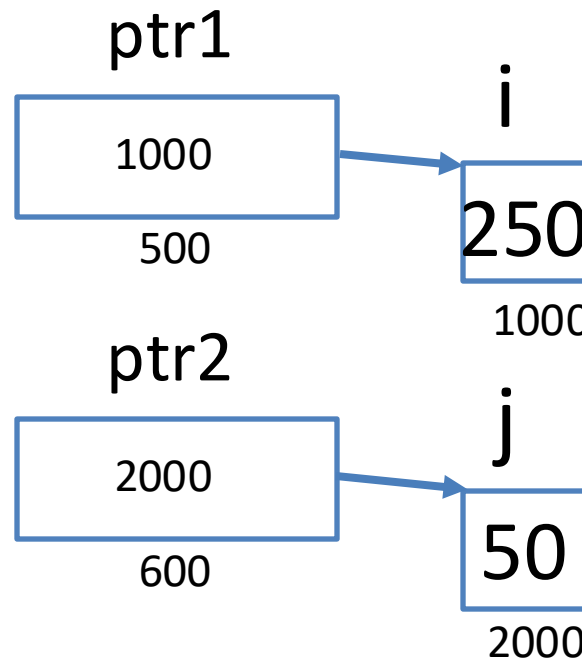 ptr2 = &j;
  *ptr2 += *ptr1;
  printf("Val = %d\n", j);

ptr1

1000

500

i

250

1000

ptr2

2000

600

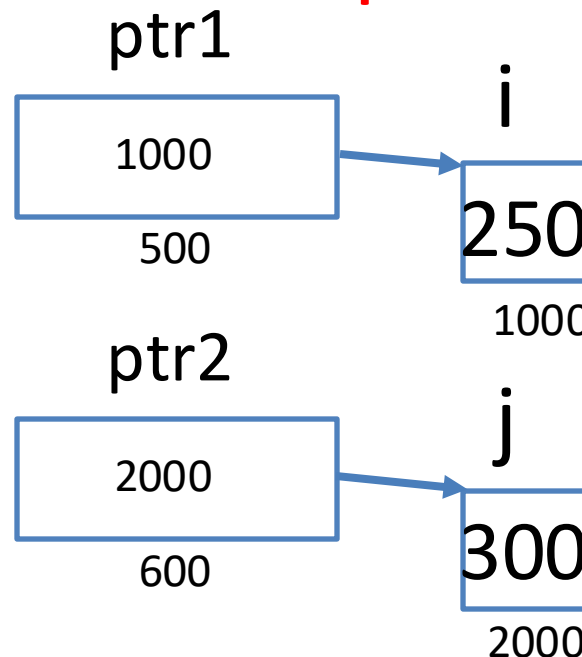j

50

2000

Q5: What is the value of j?

- int *ptr1, *ptr2, i = 10, j = 20;
  ptr1 = &i;
  *ptr1 = 150;
  ptr2 = &j;
  *ptr2 = 50;
  ptr2 = ptr1;
  *ptr2 = 250;
  ptr2 = &j;
  <mark>*ptr2 += *ptr1;</mark>
  printf("Val = %d\n", j);

*ptr2 = *ptr2 + *ptr1;
*ptr2 = 50 + 250

ptr1

1000
500

i

250
1000

ptr2

2000
600

j

300
2000

Summary

- Array of pointers

- Pointer arithmetic (e.g., subtracting, comparing)