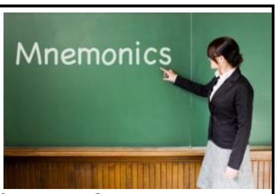


As mentioned in the textbook and class, the declaration of a pointer related variable is intended as a *mnemonic* (means 'it helps you memorize things'). For example, declaration `int *ptr;` can be interpreted as "expression `*ptr` is an `int`" -- thus `ptr` is an integer pointer.

Following this rule, what is the type that `argv` is declared to be?

```
int main(int argc, char *argv[])
{……}
```

```
int * a[]


int a[]
int ** a[]
```

64

Pointers K&R Ch 5

- Basics: Declaration and assignment (5.1)
- Pointer to Pointer (5.6)
- Pointer and functions (pass pointer by value) (5.2)
- Pointer arithmetic +- ++ -- (5.4)
- Pointers and arrays (5.3)
  - Stored consecutively
  - Pointer to array elements p + i = &a[i] *(p+i) = a[i]
  - Array name contains address of 1st element a = &a[0]
  - Pointer arithmetic on array (extension) p1-p2 p1<>!= p2
  - Array as function argument – "decay"
  - Pass sub_array
- **Array of pointers (5.6)**
- Pointer arrays vs. two dimensional arrays (5.9)
- Command line argument (5.10)
- Memory allocation (extra)
- Pointer to structures (6.4)
- Pointer to functions

Pointers K&R Ch 5

- **Pointer arrays (5.6)**
  - Declaration, initialization, accessing via element pointers
    - o Array of pointers to scalar type
    - o Array of pointers to strings

  - Pointer to the pointer arrays (what type is it?)
    - o Array of pointers to scalar type
    - o Array of pointers to strings

  - Passing pointer arrays to functions (what is it decayed to?)
    - o Array of pointers to scalar type
    - o Array of pointers to strings

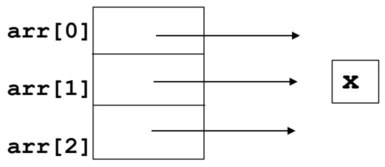
  - Pointer array vs. 2D array

Array of Pointers (5.6)

- Pointers are variables
  - Can be arrayed like others (int, char, double) …

```
int a[3]
int ** a[3]
```

```
int * arr[3]; // array of 3 pointers to integer
```



- `arr[i]` is an integer pointer `int *`

```
*arr[1] = 4
```

```
int x;
arr[1] = &x;
```

67

Precedence

Operator Type	Operator
Primary Expression Operators	() [] . ->
Unary Operators	* & + - ! ~ ++ -- (typecast) sizeof
Binary Operators	* / % arithmetic
	+ - arithmetic
	>> << bitwise
	< > <= >= relational
	== != relational
	& bitwise
	^ bitwise
	| bitwise
	&& logical
	|| logical
Ternary Operator	?:
Assignment Operators	= += -= *= /= %= >>= <<= &= ^= |=
Comma	,

```
int * arr[3]
/* array of 3
integer pointers */
```

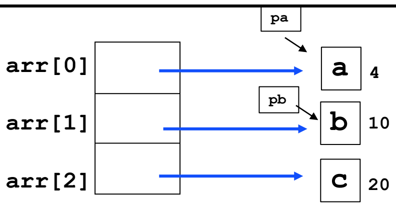
```
char * arr[5]
/* array of 5 char
pointers */
```

No () needed

```
char (*arr)[5]
/* ??? */
```

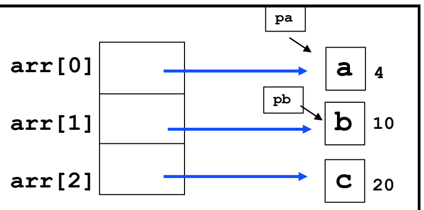
Array of pointers to scalar types

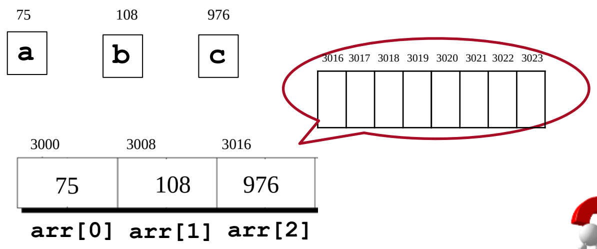
```
main(){
   int a,b,c, *pa, *pb;
   a=4; b=10;c=20;
   pa=&a, pb=&b;

   int * arr[3]; // an array of 3 (uninitialized) int pointers
   arr[0]= pa;  arr[1]= pb;  arr[2]= &c;  //different ways
   arr[0]= &a;  arr[1]= &b;
```



Array of pointers to scalar types

```
main(){
  int a,b,c, *pa, *pb;
  a=4; b=10;c=20;
  pa=&a, pb=&b;

  int * arr[3]; // an array of 3 (uninitialized) int pointers
  arr[0]= pa;  arr[1]= pb;  arr[2]= &c;  //different ways
  arr[0]= &a;  arr[1]= &b;
```





Each element is a pointer, size usually 8 bytes (regardless of the type)

Access a,b,c via arr

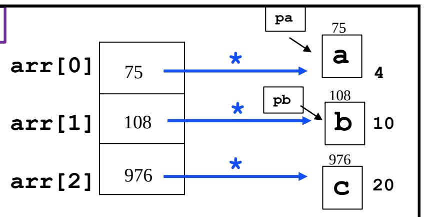
70

```
 printf("%p %p\n", arr[0], arr[1]); // 75 108
```

70

Array of pointers to scalar types

```
main(){
  int a,b,c, *pa, *pb;
  a=4; b=10;c=20;
  pa=&a, pb=&b;

  int * arr[3]; // an array of 3 (uninitialized) int pointers
  arr[0]= pa;   arr[1]= pb;  arr[2]= &c;
```



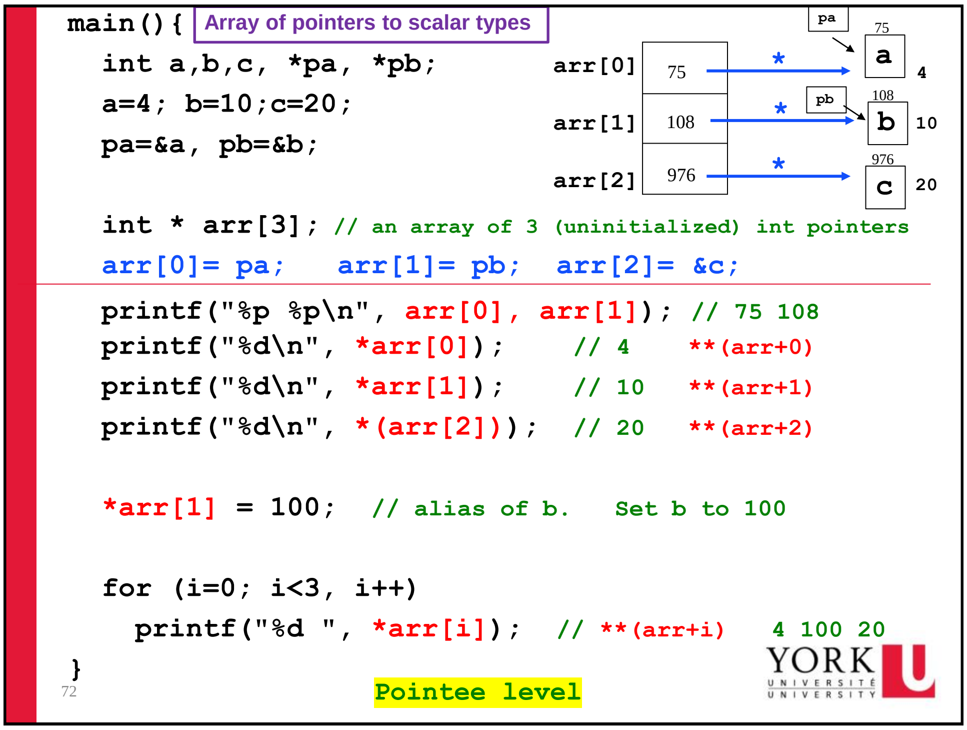
```
  printf("%p %p\n", arr[0], arr[1]); // 75 108
  printf("%d\n",                  ;  // arr[0] is a pointer to a
  printf("%d\n",                  ;  //
  printf("%d\n",                  ); //

  ?  = 100; // set b to 100
```

	Operator
ι	() [] . ->
	* & + - !~ ++ -- (typecast) sizeof

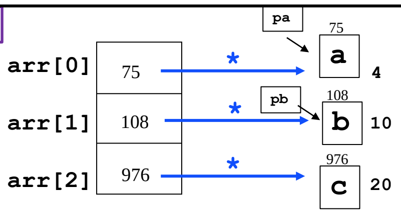
Recall:

```
int a=10;    char arr[]="apple";
int pA = &a;  char * pArr  = arr;
printf("%d %d", a, *pA);     // pointee level
printf("%s %s", arr, pArr); // pointer level
```

71



Array of pointers to scalar types



```
main(){
   int a,b,c, *pa, *pb;
   a=4; b=10;c=20;
   pa=&a, pb=&b;

   int * arr[3]; // an array of 3 (uninitialized) int pointers
   arr[0]= pa;   arr[1]= pb;   arr[2]= &c;

   printf("%p %p\n", arr[0], arr[1]); // 75 108
   printf("%d\n", *arr[0]);    // 4     **(arr+0)
   printf("%d\n", *arr[1]);    // 10    **(arr+1)
   printf("%d\n", *(arr[2])); // 20    **(arr+2)

   *arr[1] = 100;  // alias of b.   Set b to 100

   for (i=0; i<3, i++)
     printf("%d ", *arr[i]);  // **(arr+i)   4 100 20
}
```

72

Pointee level

YORK U UNIVERSITÉ UNIVERSITY