Pointers with arrays

```
#include <stdio.h>
int my_array[] = {1,23,17,4,-5,100};
int *ptr;
 int main(void)
 {    int i;
      ptr = &my_array[0]; /* point our pointer to the first element of the
      array */ printf("\n\n");
      for (i = 0; i < 6; i++)
            {
            printf("my_array[%d] = %d ",i,my_array[i]);
            printf("ptr + %d = %d\n",i, *(ptr + i));
            }
      return 0;
  }
```

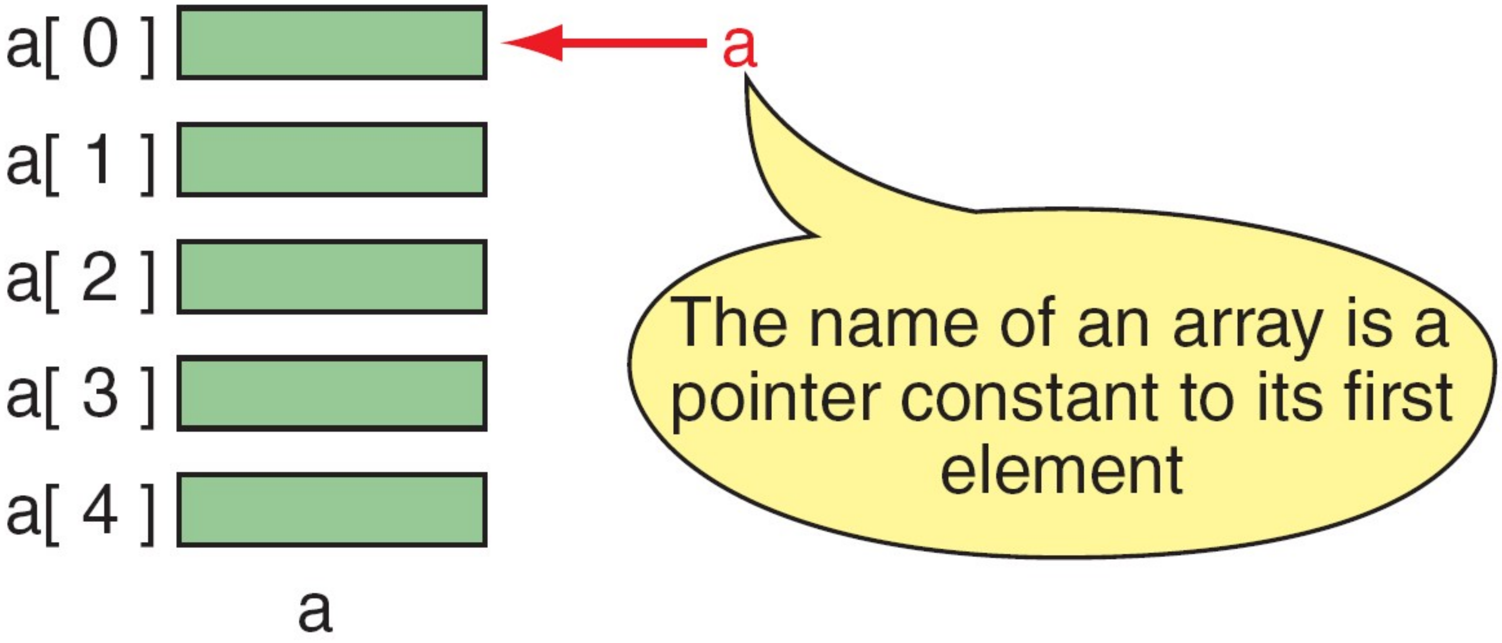
Example – 1 (strcmp with pointers)

```
#include <stdio.h>
 int main()
{         char line[20];
          char *part = "hello";
          do
          {         printf("\nEnter new String: ");
                    gets(line);
                    if(strcmp(line, part) == 0)
                              printf("The same string %s\n", line);
          } while(strlen(line)!= 0) ;
          return 0;
}
```

Example – 2 (print characters)

```
#include <stdio.h>
void printchars (char *string)
{  int count;
   for(count = 0; count < strlen(string); count ++)
   {      printf (“\n char no: %d is %c” , count , string [count]) ;
          string[count] = ‘a’ + count ;   }
 }
int main() {
   char Arr[]=“This is a test”;
   puts (Arr);
   printchars ( Arr );
   printf (“\n %s” , Arr);
  return 0; }
```

Arrays and Pointers



The name of an array is a pointer constant to its first element

Arrays and Pointers

Note

same

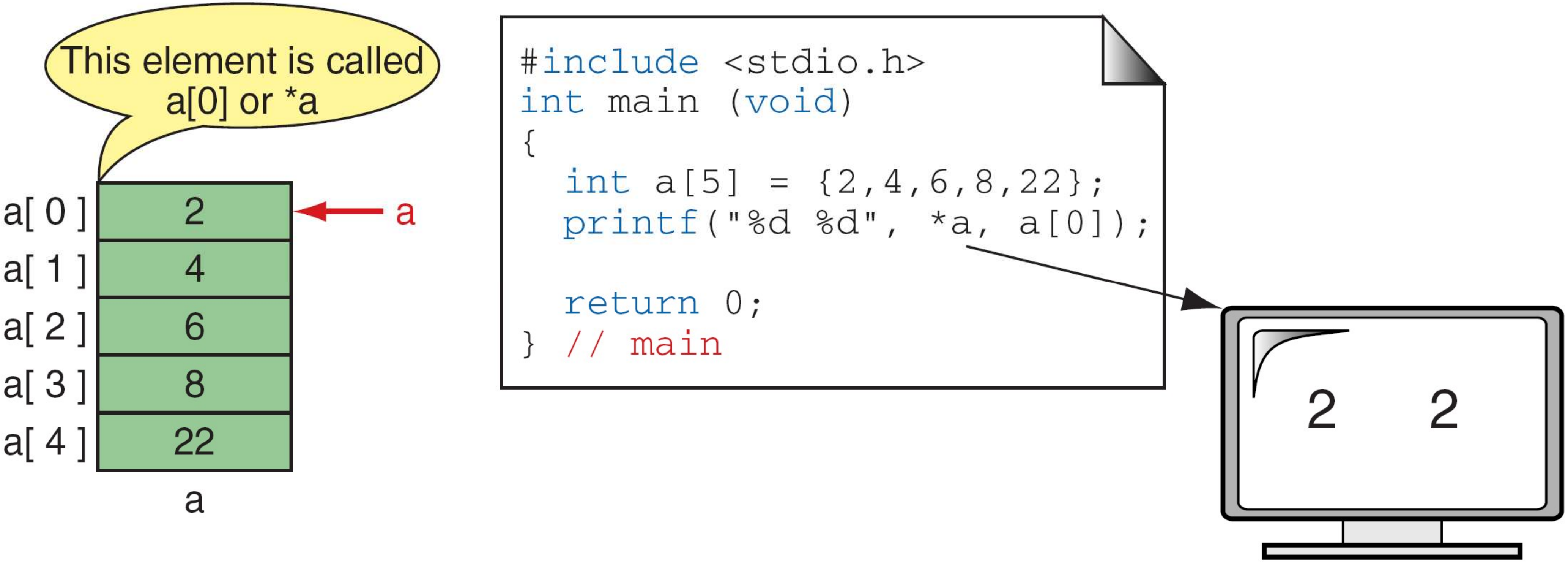
a ↔ &a[0]

a is a pointer only to the first element—not the whole array.

Note

The name of an array is a pointer constant;
it cannot be used as an *lvalue*.

Arrays and Pointers

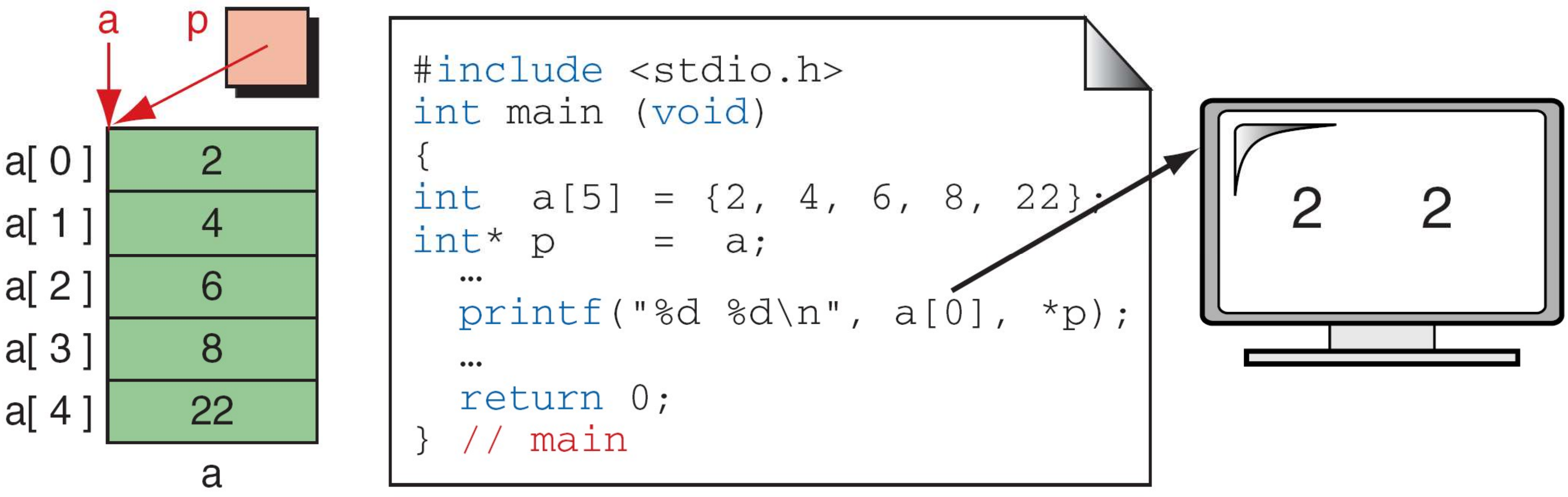


```
#include <stdio.h>
int main (void)
{
  int a[5] = {2,4,6,8,22};
  printf("%d %d", *a, a[0]);

  return 0;
} // main
```

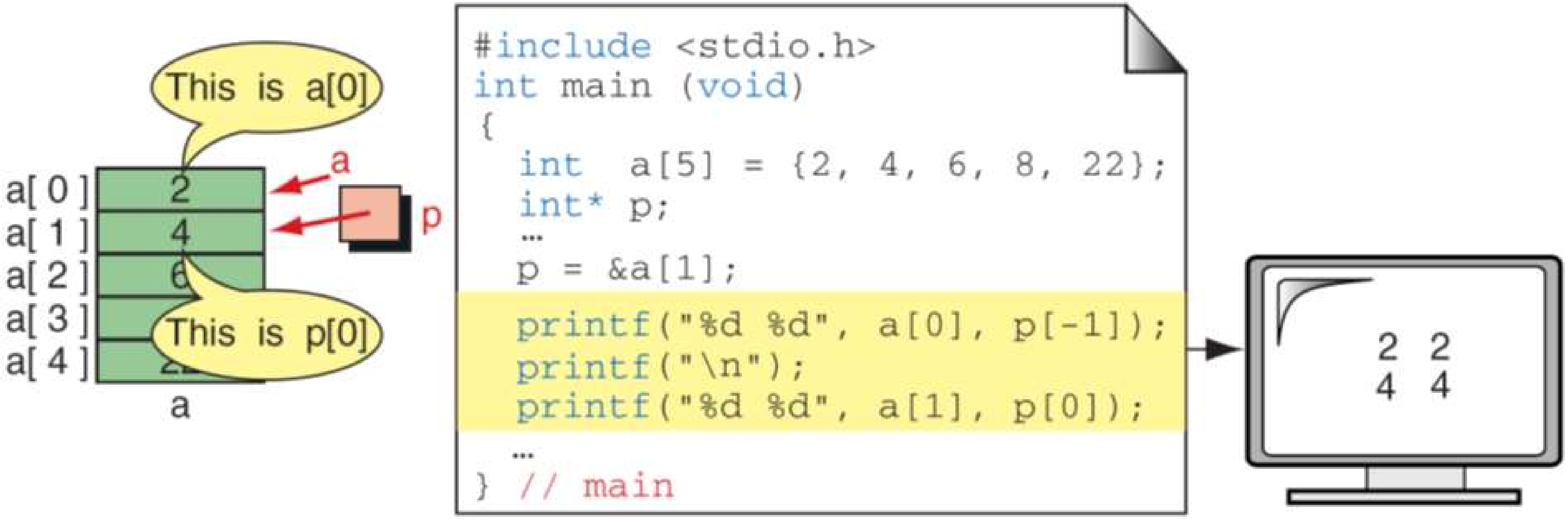
```
2   2
```

Arrays and Pointers



```
#include <stdio.h>
int main (void)
{
int  a[5] = {2, 4, 6, 8, 22};
int* p    =  a;
   …
   printf("%d %d\n", a[0], *p);
   …
   return 0;
} // main
```

Arrays and Pointers



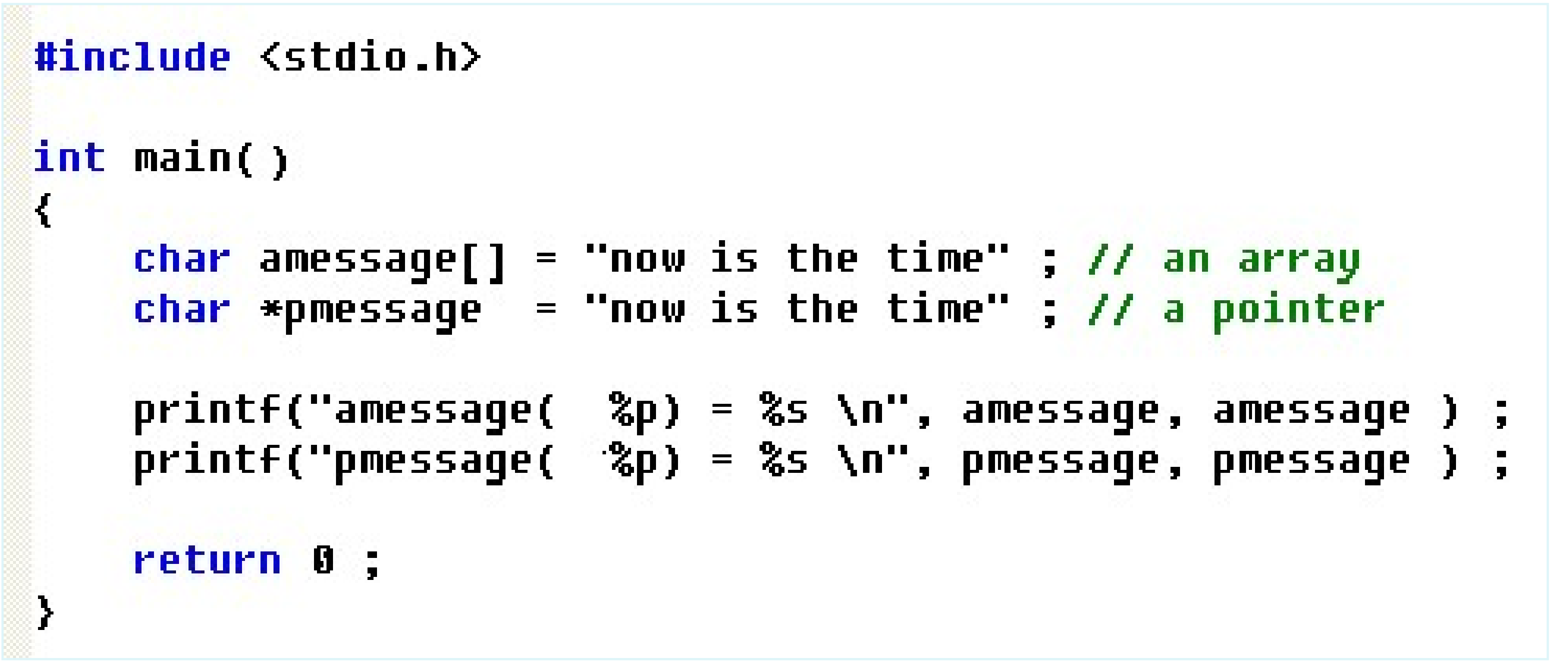
```
#include <stdio.h>
int main (void)
{
   int  a[5] = {2, 4, 6, 8, 22};
   int* p;
   …
   p = &a[1];
   printf("%d %d", a[0], p[-1]);
   printf("\n");
   printf("%d %d", a[1], p[0]);
   …
} // main
```

```
2 2
4 4
```

Note

To access an array, any pointer to the first element can be used instead of the name of the array.

Pointers and Arrays



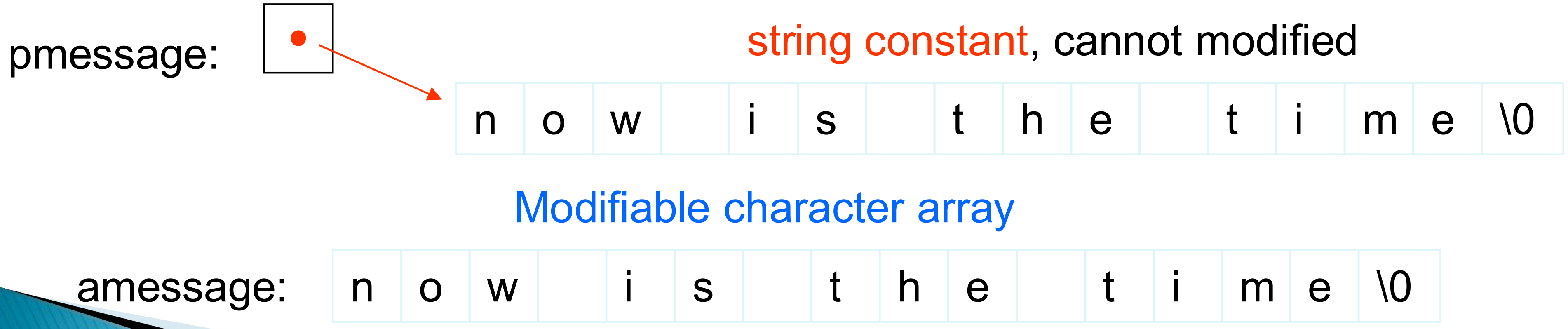
```
#include <stdio.h>

int main( )
{
    char amessage[] = "now is the time" ; // an array
    char *pmessage  = "now is the time" ; // a pointer

    printf("amessage(  %p) = %s \n", amessage, amessage ) ;
    printf("pmessage(  %p) = %s \n", pmessage, pmessage ) ;

    return 0 ;
}
```

Compiler determine length of string and then size of amessage



pmessage:

string constant, cannot modified

n	o	w		i	s		t	h	e		t	i	m	e	\0

Modifiable character array

amessage:

n	o	w		i	s		t	h	e		t	i	m	e	\0

Example – 3 (Array and pointers)

```
#include <stdio.h>
int my_array[] = {1,23,17,4,-5,100};
int *ptr;
 int main(void)
 {    int i;
    ptr = &my_array[0]; /* point our pointer to the first element of
    the array */ printf("\n\n");
    for (i = 0; i < 6; i++)
         {         printf("my_array[%d] = %d ", i , my_array[i] );
                   printf(“\t ptr + %d = %d\n", i , *(ptr + i) );
         }
    return 0;
  }
```

Example – 4 (count a character)

```
#include <stdio.h>
int countnchar (char *string, char ch)
{  char *p;
   int count = 0;
   for(p = string; *p != '\0'; p++)
   {      if(*p == ch)     count++;          }
   return count; }
int main() {
  char f = ‘A’; int x ;
  char Arr[]=“This is A test for letter A”;
  x = countnchar ( Arr , f );
  printf (“Letter A exist : %d times”, x );
  return 0; }
```

Example – 5 (swap with array index)

```
#include <stdio.h>
# define N 5
void swap (int *a, int *b){
  int temp = *a;
  *a = *b;
  *b = temp;              }
int main() {
  int i, j;
  int Arr[N]={1,2,3,4,5};
  for (i=0; i < N/2 ; i++)
      swap ( & Arr [ i ] , & Arr [ (N–1) – i ] );
  return 0; }
```

Example – 6 (swap with pointers)

```
#include <stdio.h>
# define N 5
void swap (int *a, int *b){
  int temp = *a;
  *a = *b;
  *b = temp;              }
int main() {
  int i, j;
  int Arr[N]={1,2,3,4,5};
  for (i=0; i < N/2 ; i++)
    swap ( Arr + i , Arr + (N–1) – i );
  return 0; }
```

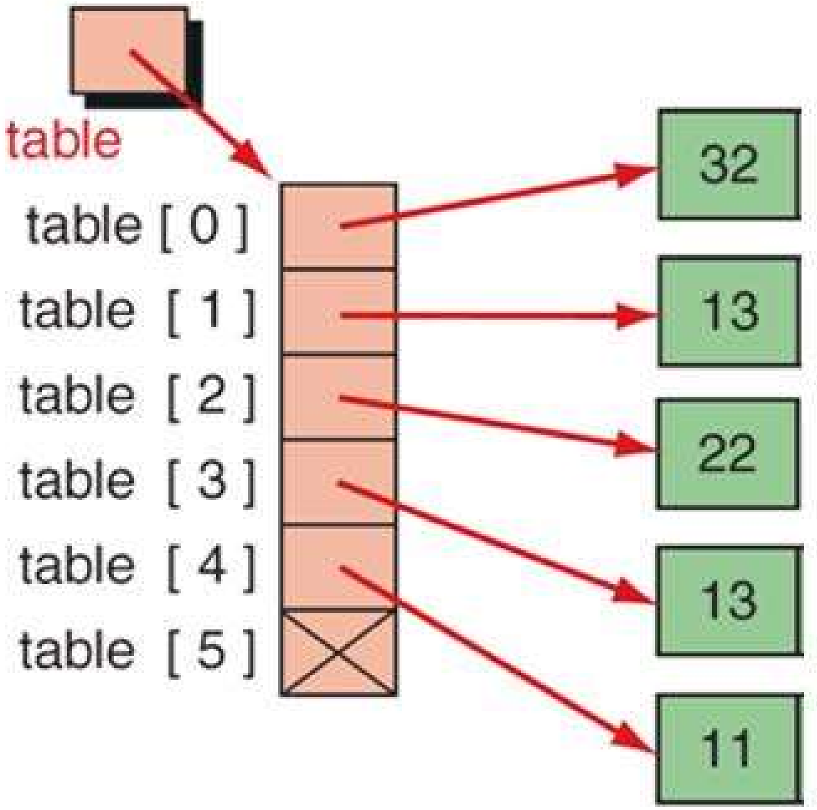
Example – 7 (reverse with pointers)

```
#include <stdio.h>
void reverse(char *mysrt)
{  char * lp = mysrt;                        /* left pointer */
   char *rp = &mysrt[strlen(mysrt)-1];          /* right
   pointer */
   char tmp;
   while(lp < rp)    {
        tmp = *lp;
        *lp = *rp;
        *rp = tmp;
        lp++;
        rp--;          }
}
```

```
int main()
{  char Arr[]="This is a test";
    puts(Arr);
    reverse (Arr);
     printf("\n");
     puts(Arr);
     return 0;
}
```

Arrays of Pointers

- Arrays can contain pointers to …..



Array of Pointers

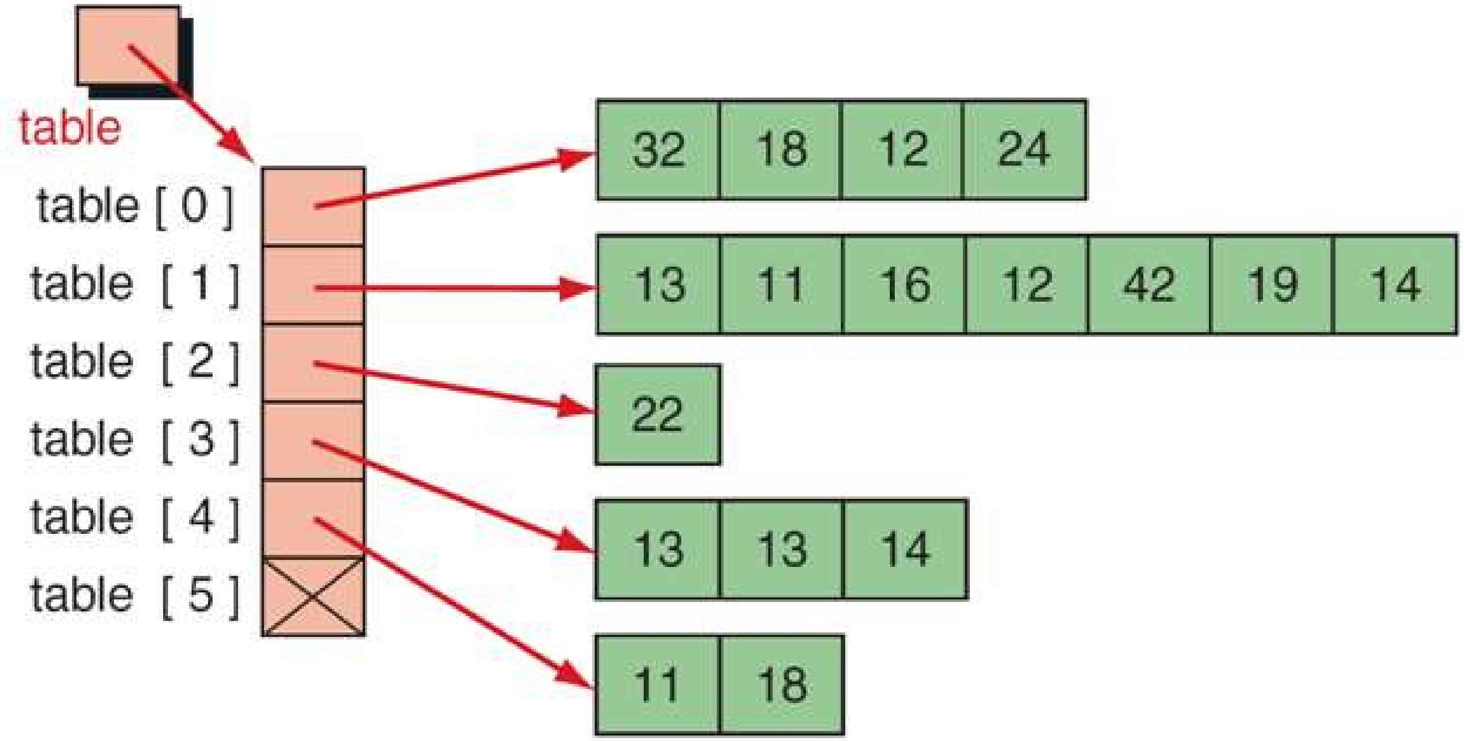
```
int  x = 4;
int *y = &x;
int *z[4] = {NULL, NULL, NULL, NULL};
int  a[4] = {1, 2, 3, 4};

z[0] = a;          // same as &a[0];
z[1] = a + 1;      // same as &a[1];
z[2] = a + 2;// same as &a[2];
z[3] = a + 3;// same as &a[3];

for (x=0;x<4;x++)
 printf("\n %d --- %d ",a[x],*z[x]);
```

Arrays of Pointers

- Arrays can contain pointers to (**array**)



Array of Pointers

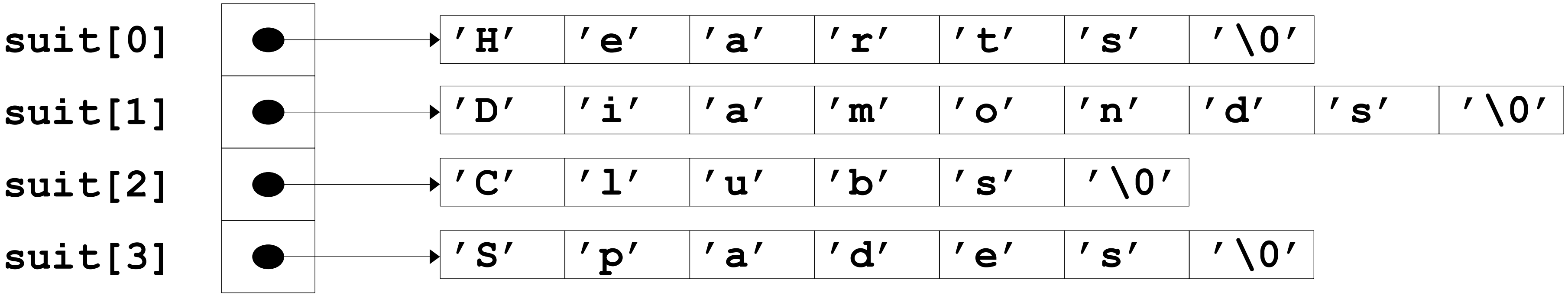
- For example: an array of strings

```
char *suit[ 4 ] = {        "Hearts",
                        "Diamonds",
                        "Clubs",
                        "Spades" };
```

  - Strings are pointers to the first character
  - **`Each`** element of **`suit`** is a pointer to a **`char`**
  - The strings are not actually stored in the array **`suit`**, only pointers to the strings are stored
  - **`suit`** array has a fixed size, but strings can be of any size

Arrays of Pointers

- **`char *`** – each element of **`suit`** is a pointer to a **`char`**
- The strings are not actually stored in the array **`suit`**, only pointers to the strings are stored
- **`suit`** array has a fixed size, but strings can be of any size



suit[0]	●	'H'	'e'	'a'	'r'	't'	's'	'\0'	
suit[1]	●	'D'	'i'	'a'	'm'	'o'	'n'	'd'	's'	'\0'
suit[2]	●	'C'	'l'	'u'	'b'	's'	'\0'			
suit[3]	●	'S'	'p'	'a'	'd'	'e'	's'	'\0'		

Example – 8 (Array of strings)

```
char *suit[ 4 ] = { "Hearts",
                    "Diamonds",
                    "Clubs",
                    "Spades" };
int main()
{ int  x ;
  for (x = 0; x < 4 ; x++)
    printf("\n %s   ---   %d ",
           suit[x],strlen(suit[x]));
return 0; }
```

Pointer to Structure

- We can use pointer to struct:
  - `struct MyPoint {int x, int y};`
  - `MyPoint point, *ptr;`
  - `point.x = 0;`
  - `point.y = 10;`
  - `ptr = &point;`
  - **`ptr->x = 12;`** same as **`(*ptr).x`**
  - **`ptr->y = 40;`** same as **`(*ptr).y`**