Array name

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact 1

The name of the array stores the address of the first element.

`&a[0]` is same as `a`.

Fact 2

The name of the array is not a variable.

`a` is this sense is constant.

Array index

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

```
printf("%d", a[2]);
```

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

`printf("%d", a[2]);` or `printf("%d", *(a + 2);`

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

It applies to pointers as well.

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

It applies to pointers as well.

```
int *p;
```

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

It applies to pointers as well.

```
int *p;
p = a;
```

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

It applies to pointers as well.

```
int *p;
p = a;
printf("%d", p[2]);
```

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

It applies to pointers as well.

```
int *p;
p = a;
printf("%d", p[2]);
```

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

It applies to pointers as well.

```
int *p;
p = a;
printf("%d", p[2]); → 3
```

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

It applies to pointers as well.

```
int *p;
p = a;
```

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

It applies to pointers as well.

```
int *p;
p = a;
for (i = 0; i < 3; i++) {

}
```

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

It applies to pointers as well.

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

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

It applies to pointers as well.

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

Array index

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Fact

`a[i]` is same as `*(a + i)`.

It applies to pointers as well.

```
int *p;
p = a;
for (i = 0; i < 3; i++) {
  printf("%d", p[i]); → 1 2 3
}
```

Pointer subtraction

Pointer subtraction

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;
```

Pointer subtraction

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;

int *p = a;
```

Pointer subtraction

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;

int *p = a;
int *q = &a[2];
```

Pointer subtraction

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;

int *p = a;
int *q = &a[2];
printf("%d", q - p);
```

Pointer subtraction

```
int a[3];
a[0] = 1, a[1] = 2, a[2] = 3;

int *p = a;
int *q = &a[2];
printf("%d", q - p); → 2
```

Definition

Gives the number of array elements between the two address.

Strings

Strings

Definition

Strings

Definition

- Array of characters.

Strings

Definition

- Array of characters.

```
char s[10];
```

Strings

Definition

- Array of characters.

```
char s[10];
```

- Ends with the null character – `'\0'`.

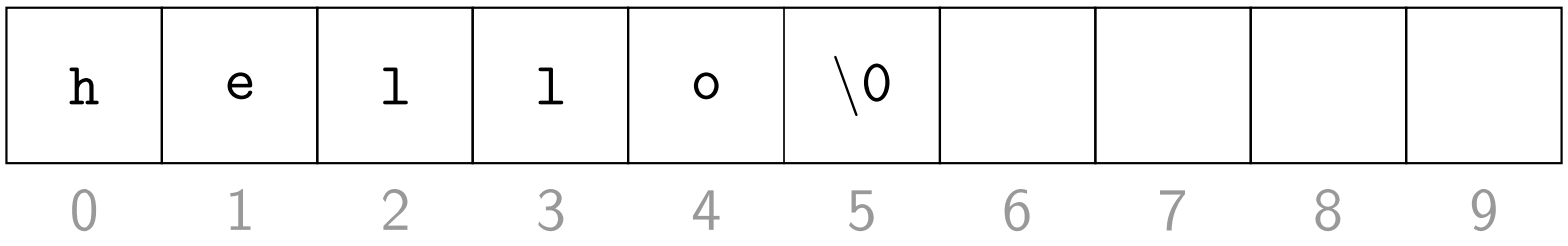
Strings

Definition

- Array of characters.

```
char s[10];
```

- Ends with the null character – '\0'.



Strings

Initialisation

Strings

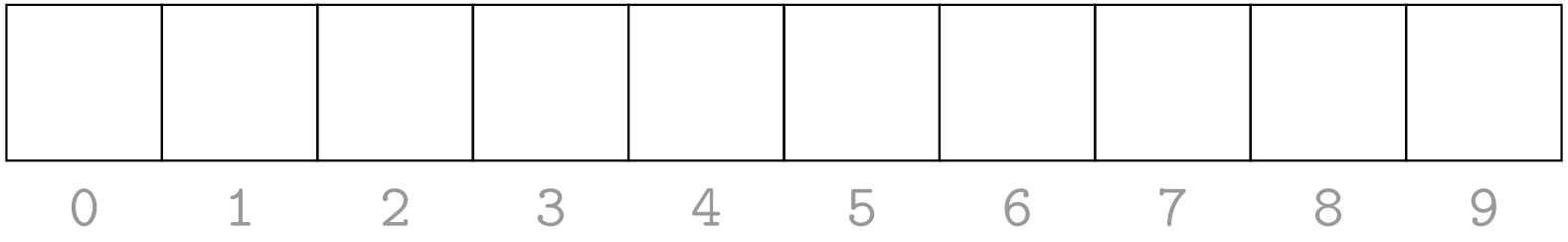
Initialisation

```
char s[10];
```

Strings

Initialisation

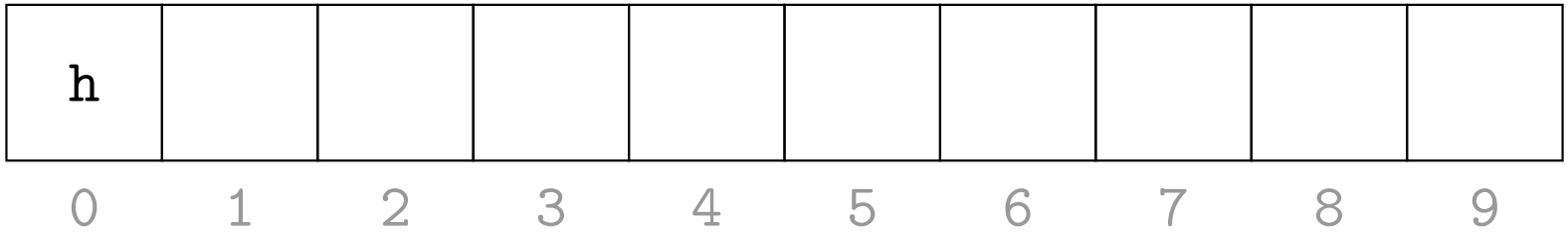
```
char s[10];
```



Strings

Initialisation

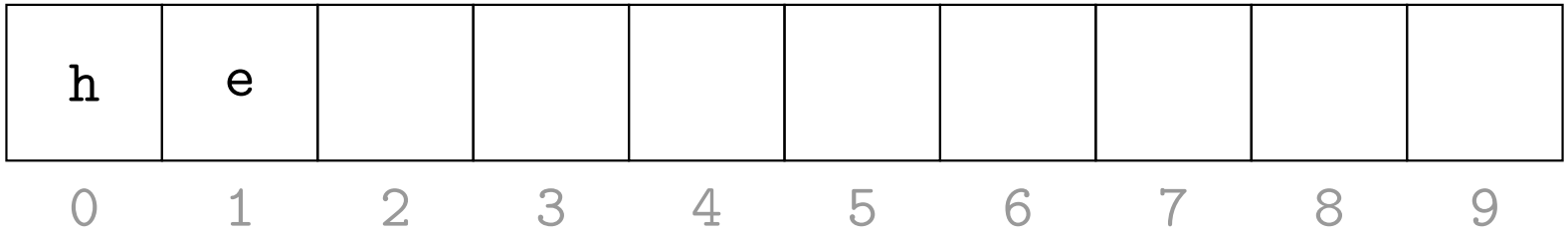
```
char s[10];
s[0] = 'h';
```



Strings

Initialisation

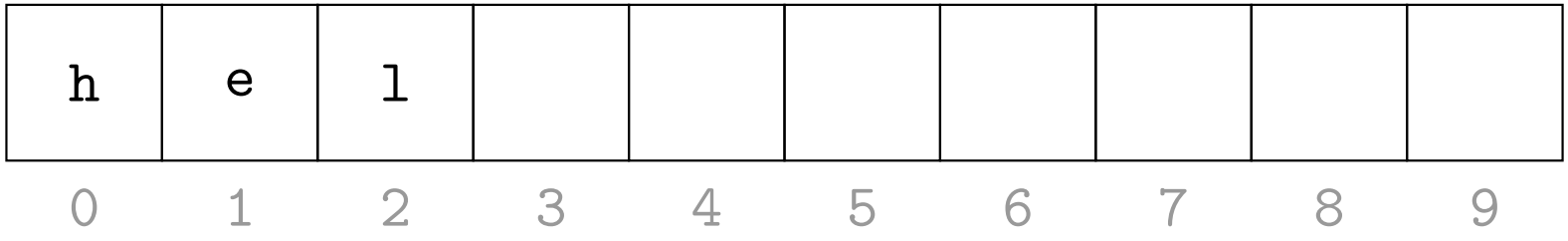
```
char s[10];
s[0] = 'h';
s[1] = 'e';
```



Strings

Initialisation

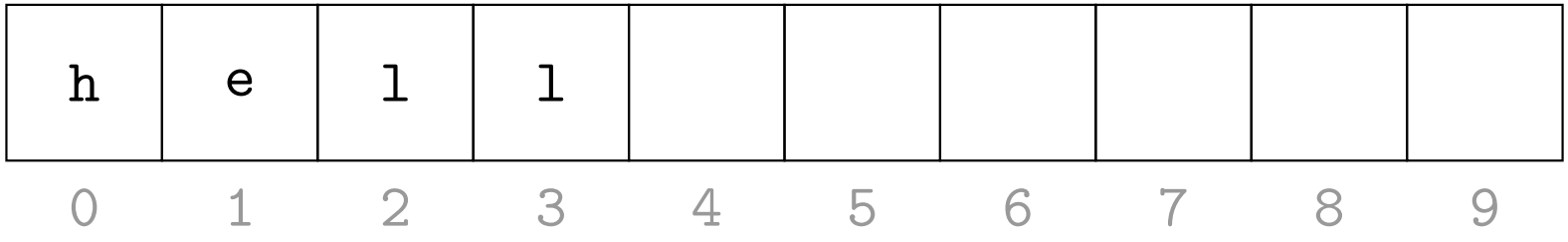
```
char s[10];
s[0] = 'h';
s[1] = 'e';
s[2] = 'l';
```



Strings

Initialisation

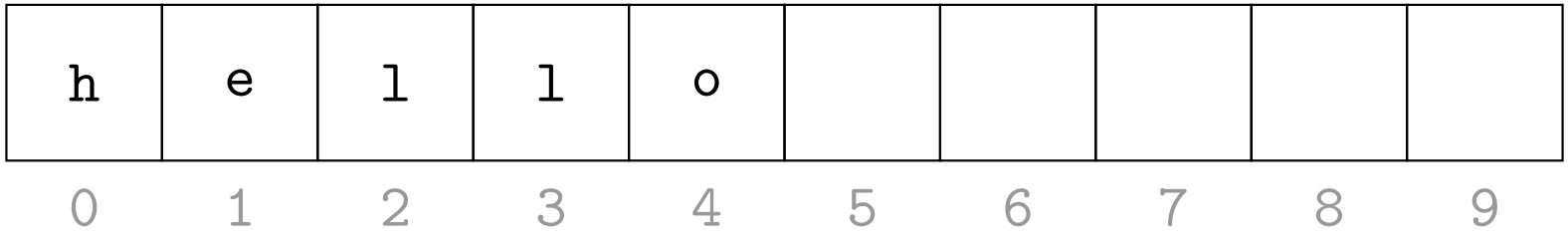
```
char s[10];
s[0] = 'h';
s[1] = 'e';
s[2] = 'l';
s[3] = 'l';
```



Strings

Initialisation

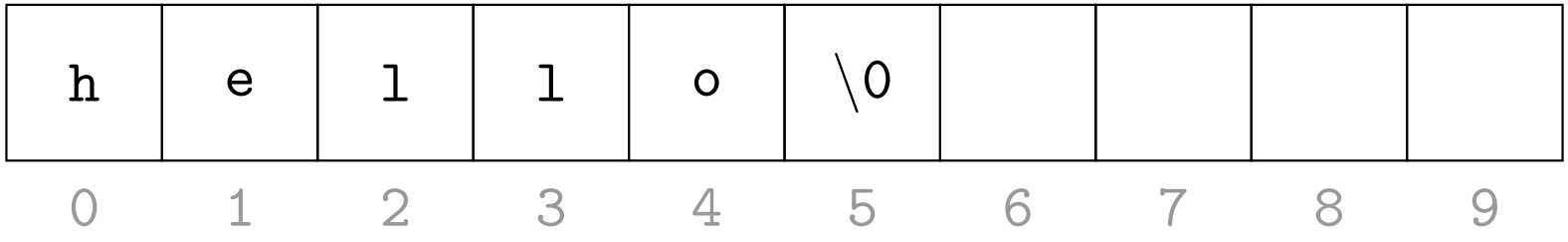
```
char s[10];
s[0] = 'h';
s[1] = 'e';
s[2] = 'l';
s[3] = 'l';
s[4] = 'o';
```



Strings

Initialisation

```
char s[10];
s[0] = 'h';
s[1] = 'e';
s[2] = 'l';
s[3] = 'l';
s[4] = 'o';
s[5] = '\0';
```



Strings

Initialisation

```
char s[10];
```

Strings

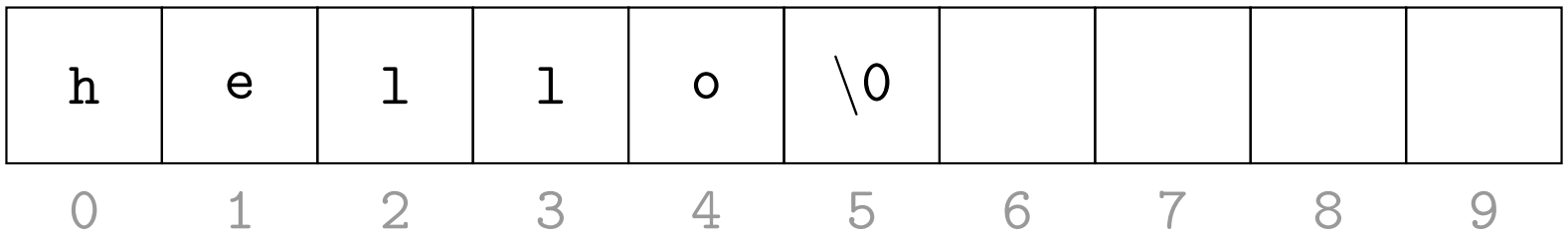
Initialisation

```
char s[10] = {'h','e','l','l','o','\0'};
```

Strings

Initialisation

```
char s[10] = {'h','e','l','l','o','\0'};
```



Strings

Initialisation

Strings

Initialisation

```
char s[]
```

Strings

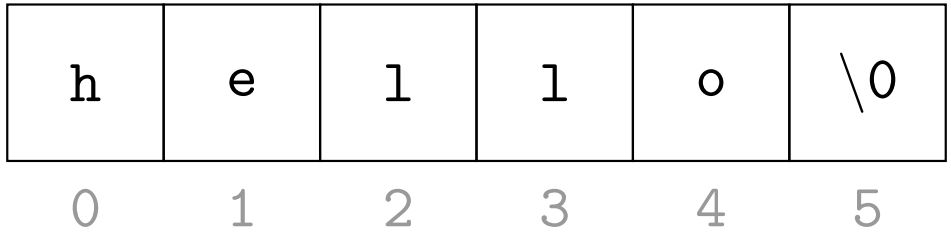
Initialisation

```
char s[] = "hello";
```

Strings

Initialisation

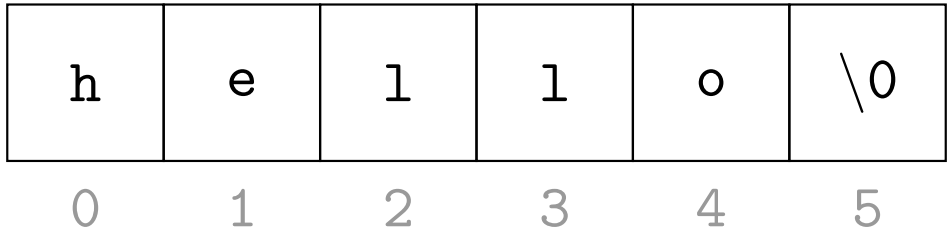
```
char s[] = "hello";
```



Strings

Initialisation

```
char s[] = "hello";
```

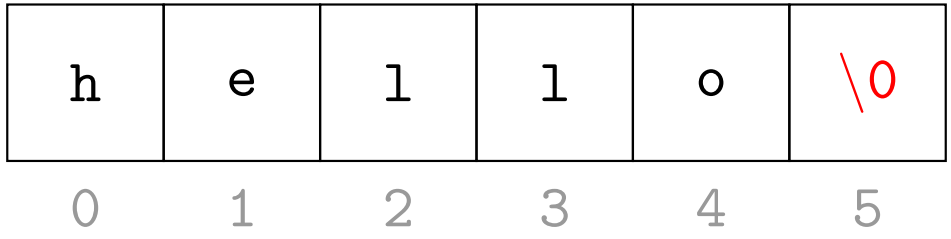


Comments

Strings

Initialisation

```
char s[] = "hello";
```



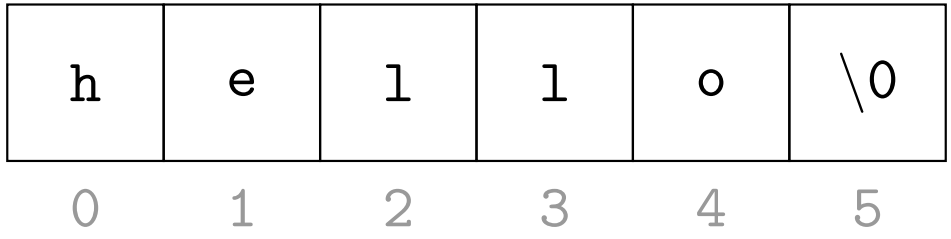
Comments

- Null character is added automatically.

Strings

Initialisation

```
char s[] = "hello";
```



Comments

- Null character is added automatically.
- Size of array is one more than the number of characters.

Strings

Initialisation

Strings

Initialisation

```
char s[]
```

Strings

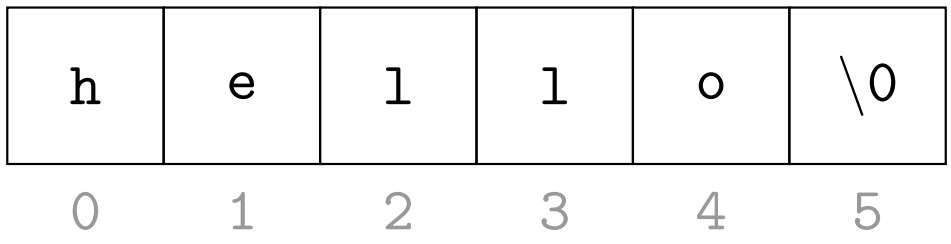
Initialisation

```
char s[] = {'h','e','l','l','o','\0'};
```

Strings

Initialisation

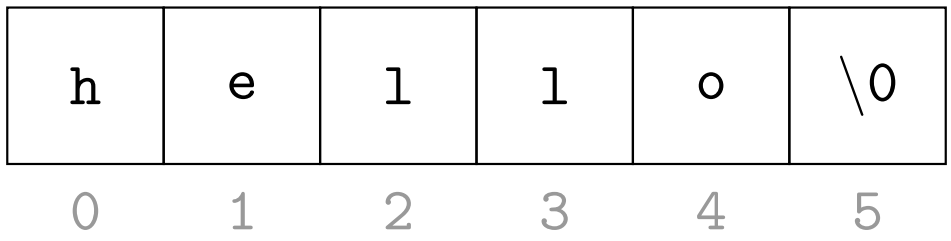
```
char s[] = {'h','e','l','l','o','\0'};
```



Strings

Initialisation

```
char s[] = {'h','e','l','l','o','\0'};
```



Comments

- Size of array is equal to the number of characters.

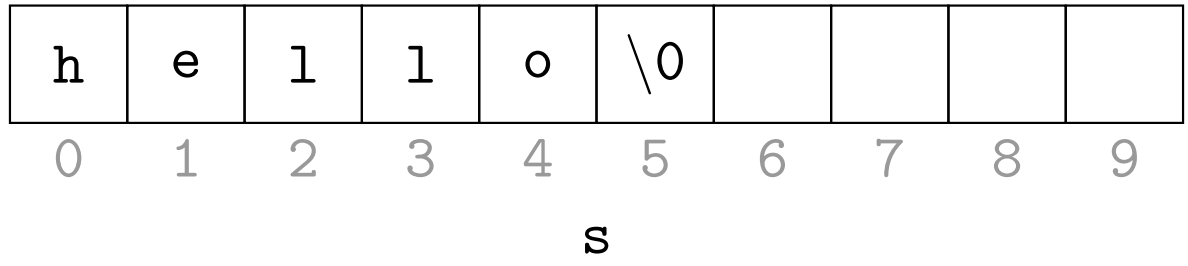
String Length

String Length

```
char s[10] = "hello";
```

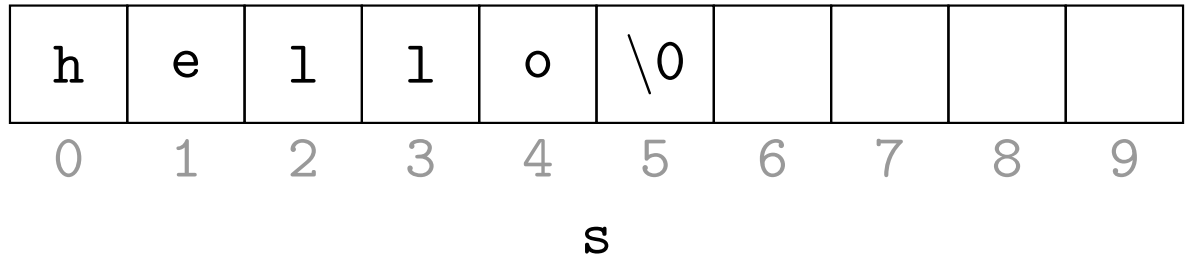
String Length

```
char s[10] = "hello";
```



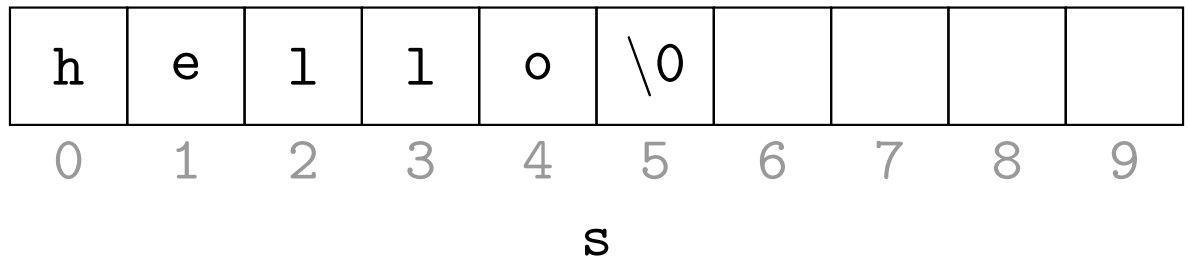
String Length

```
char s[10] = "hello";
int nc; /* number of characters */
```



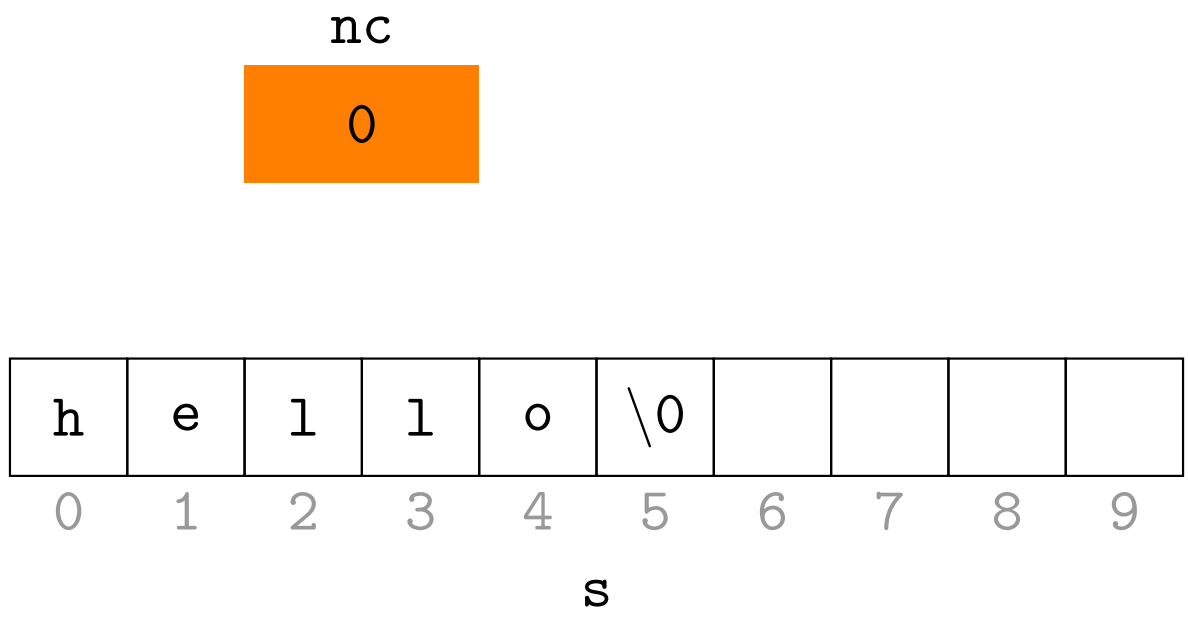
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
```



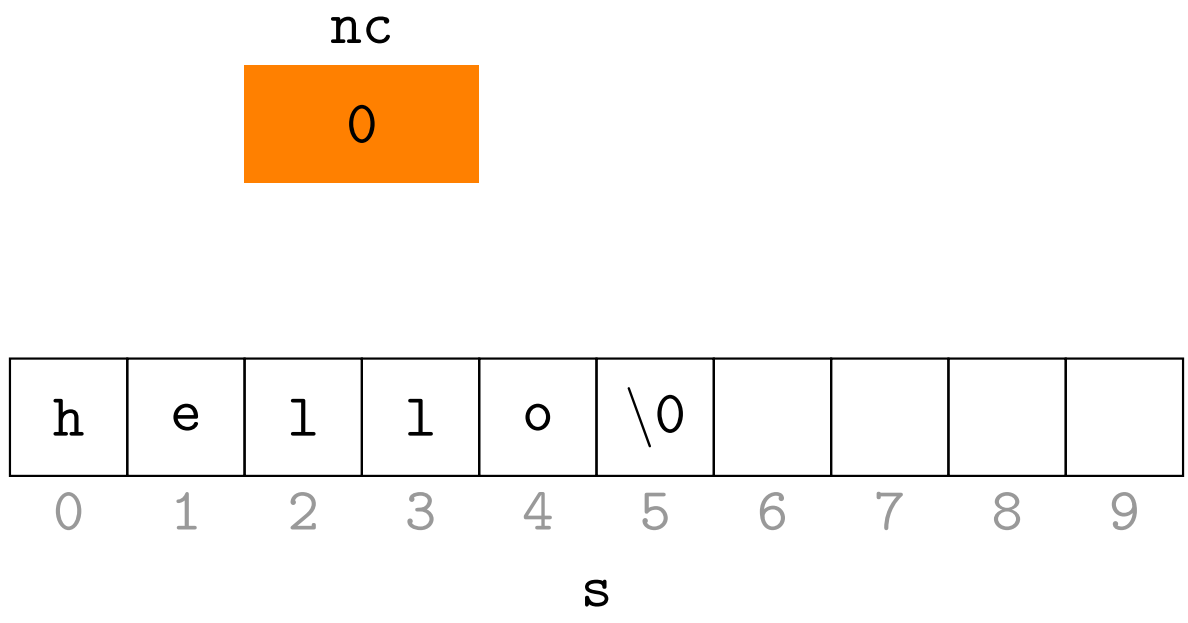
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
```



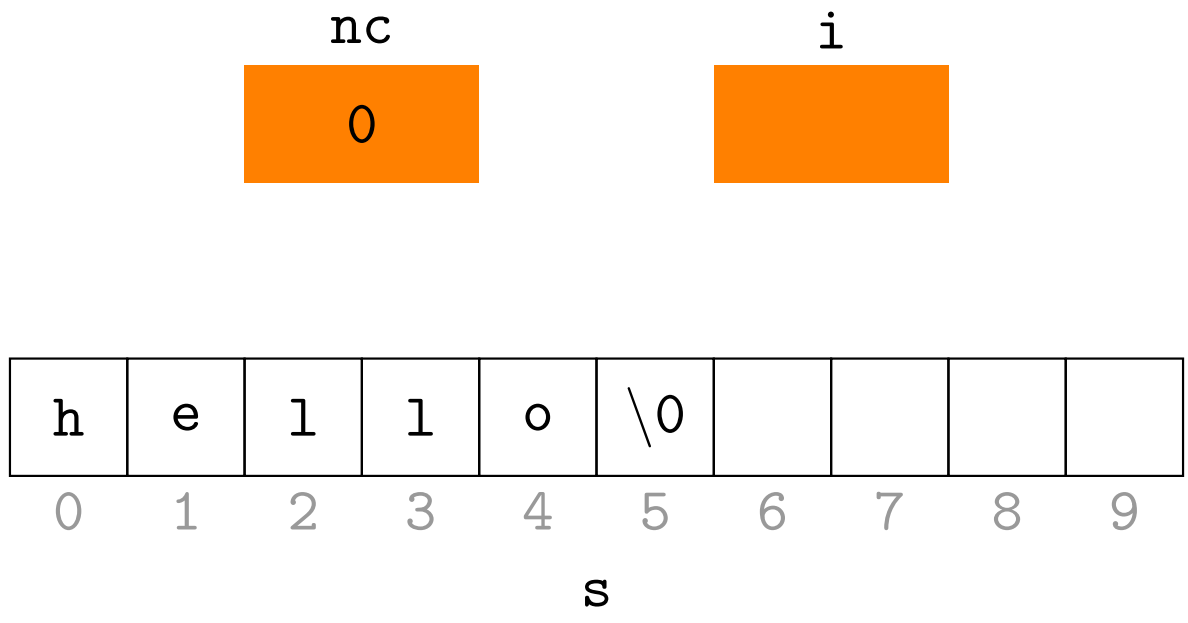
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
```



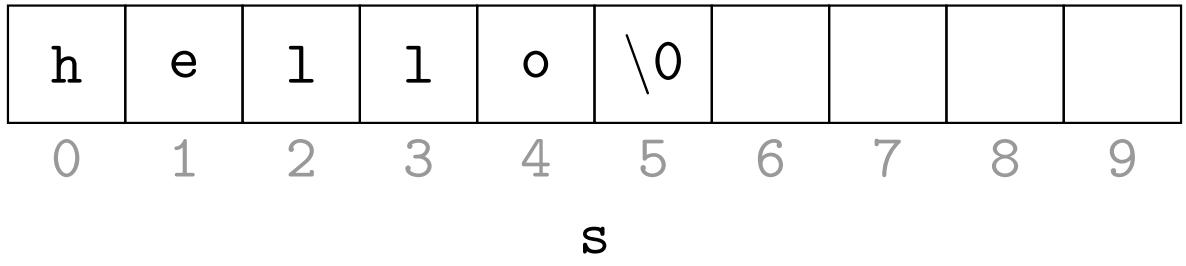
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
```



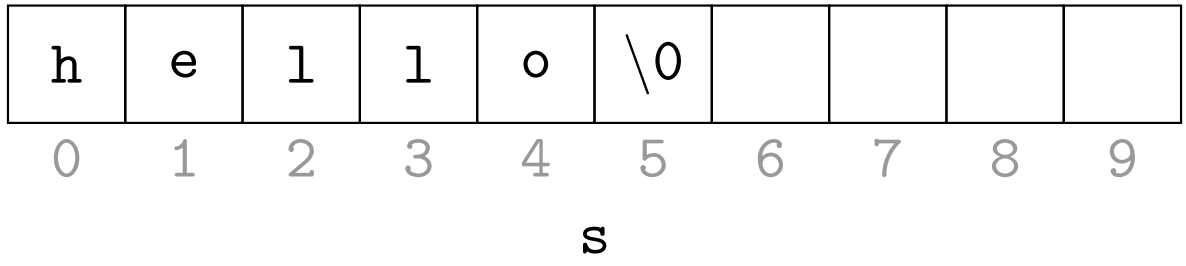
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
for (     ;                ;    ) {

}
```



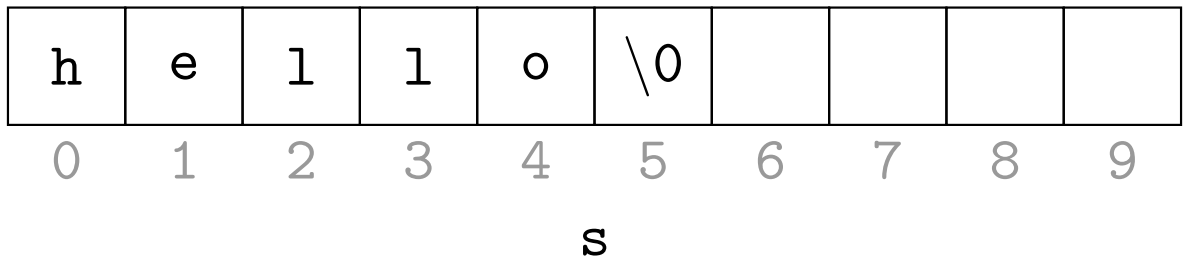
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0;                ;    ) {

}
```



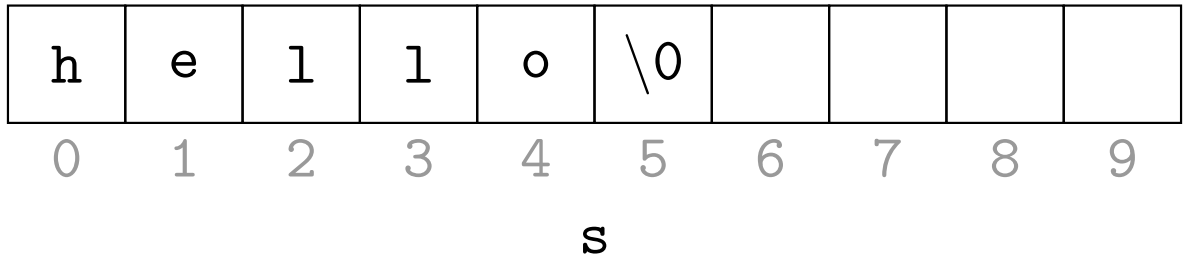
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0';    ) {

}
```



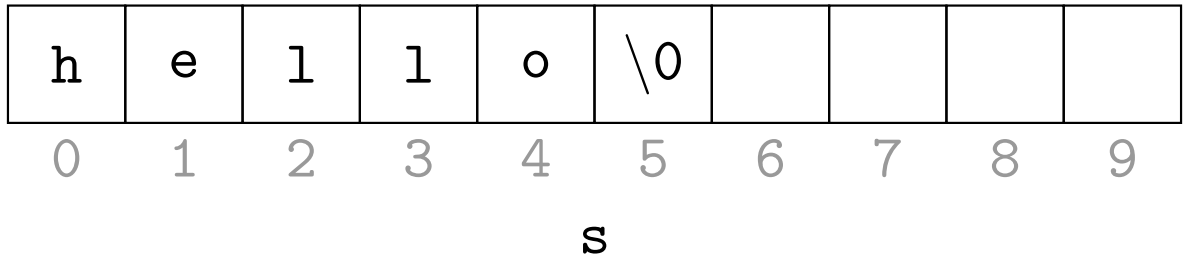
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0';    ) {
  nc++;
}
```



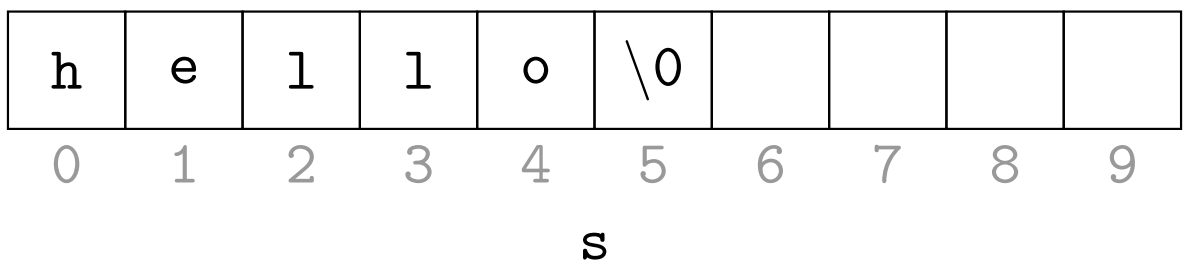
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```



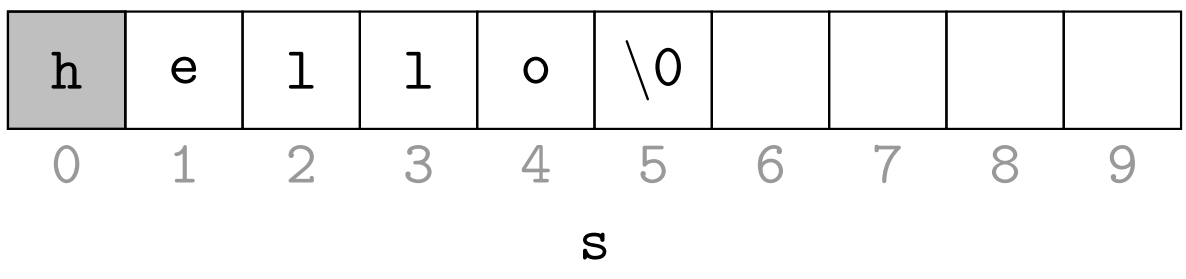
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```



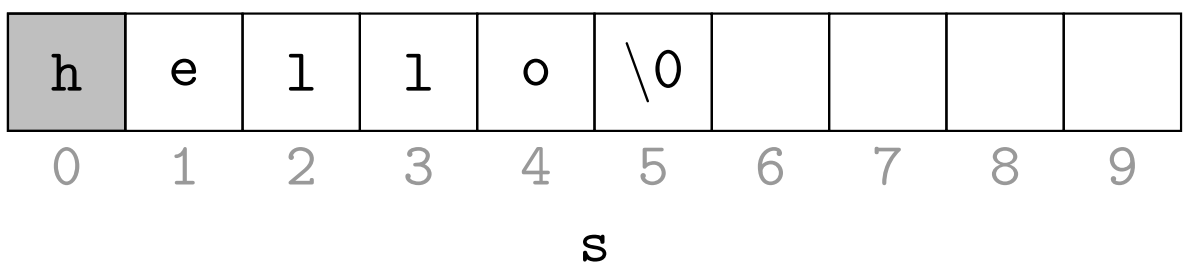
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```



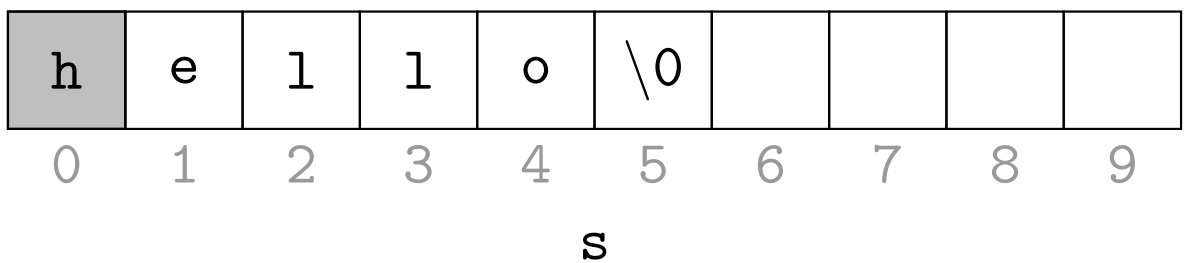
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```



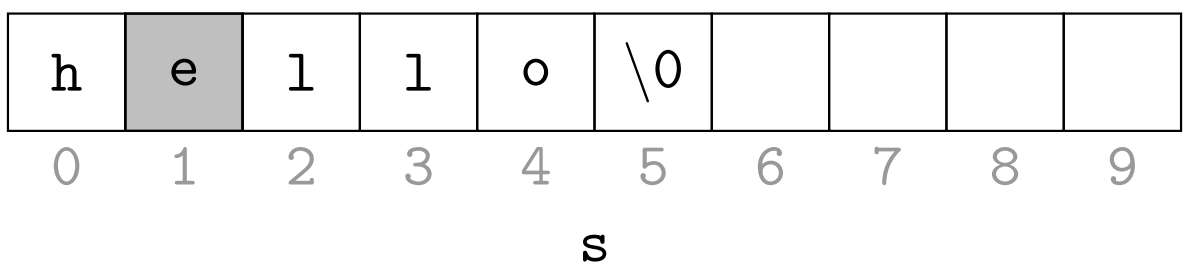
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```



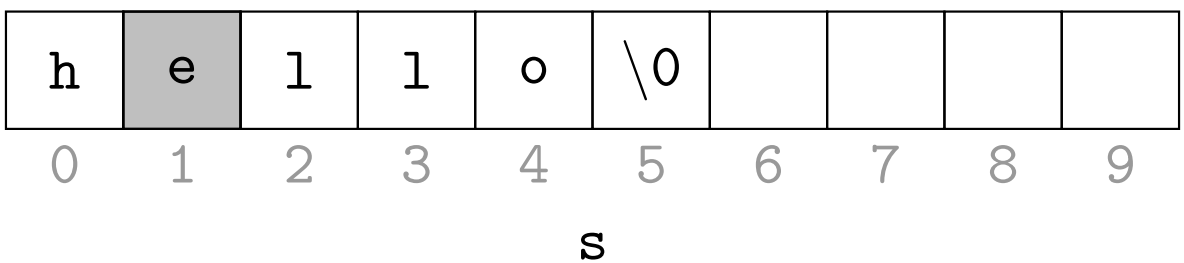
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```



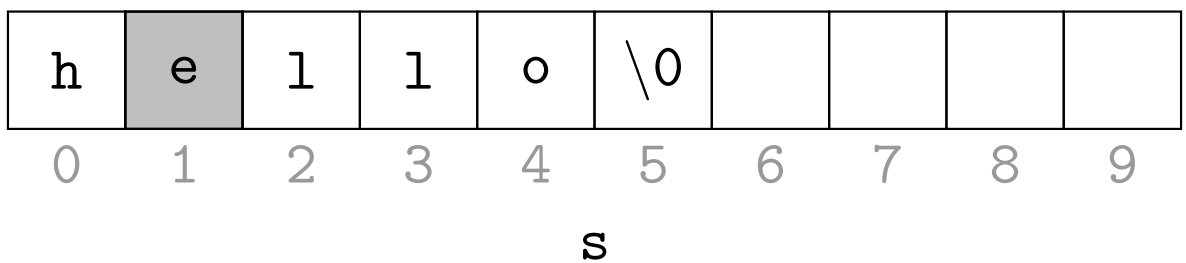
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```



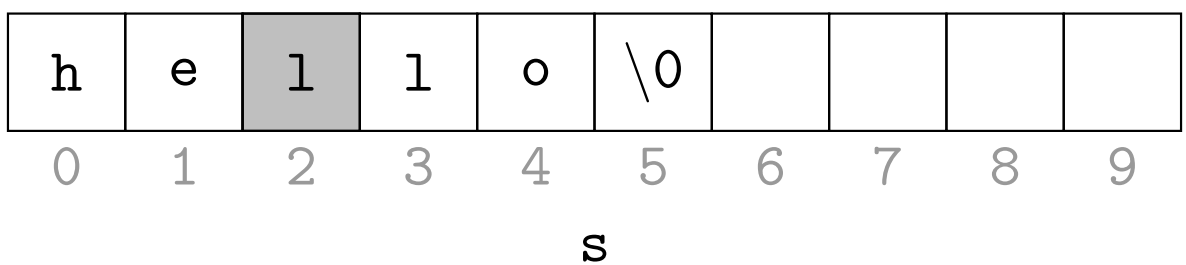
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```



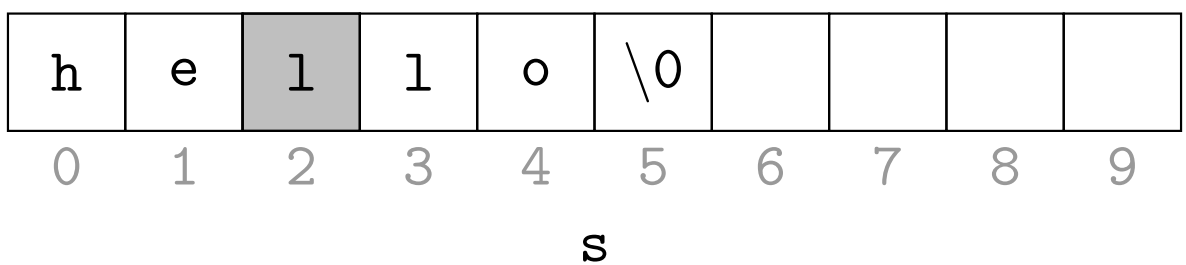
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```



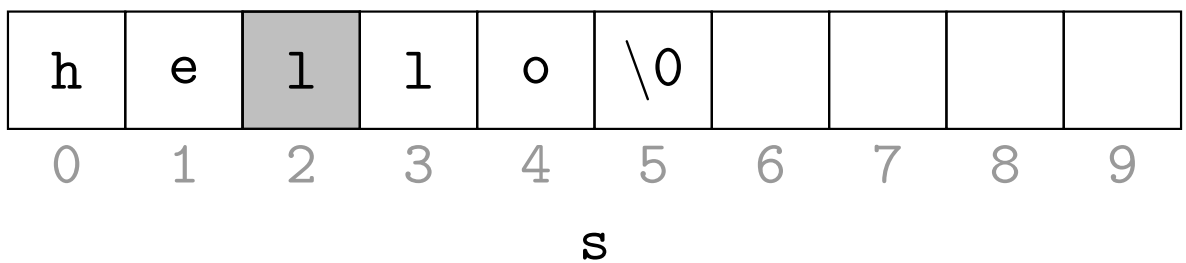
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```



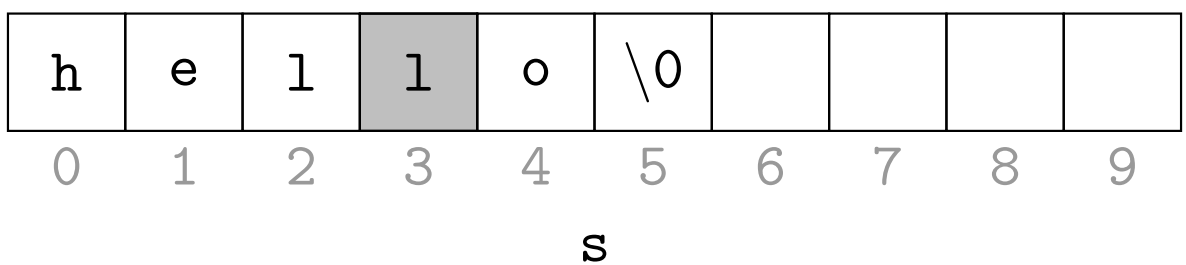
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```



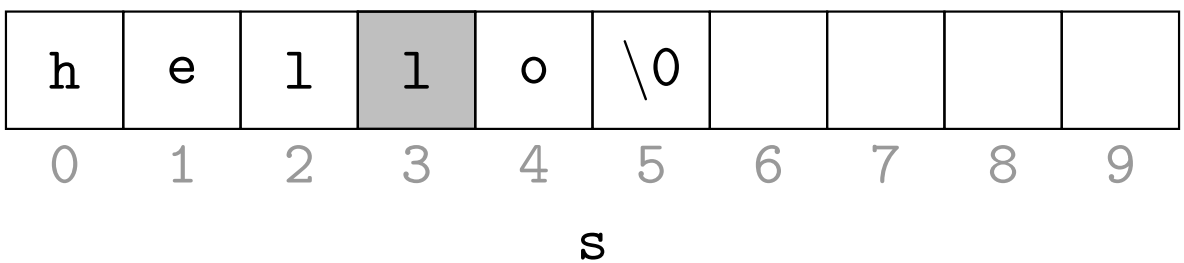
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```



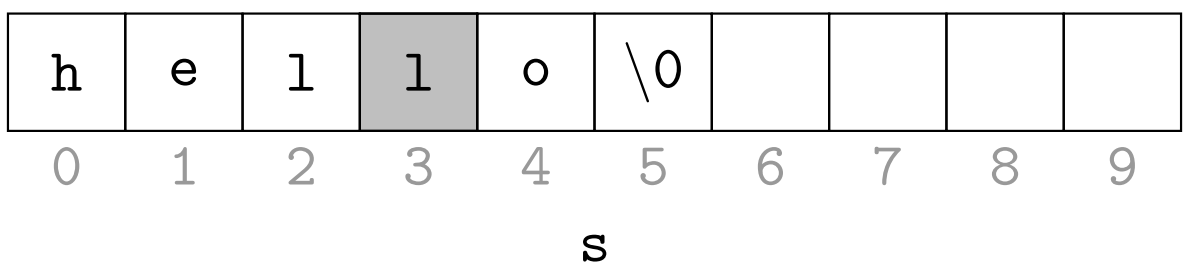
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```



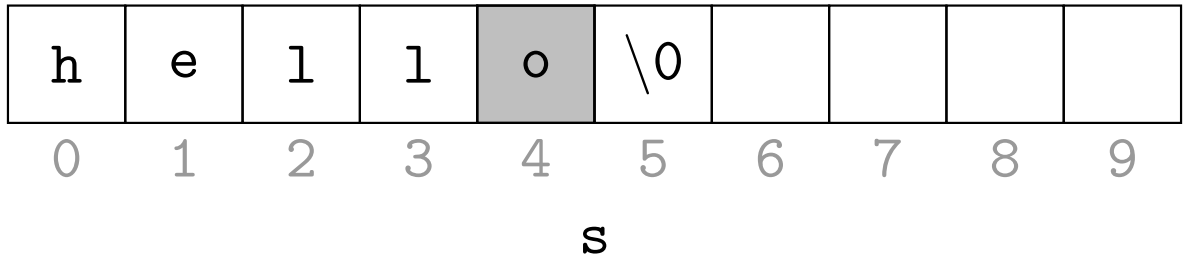
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```



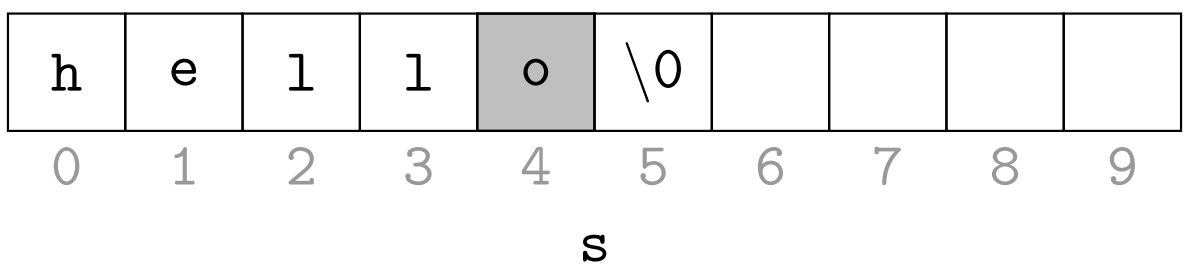
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```



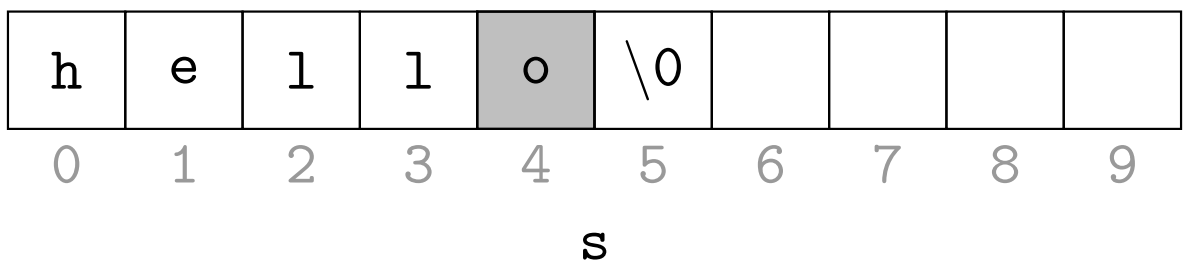
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```



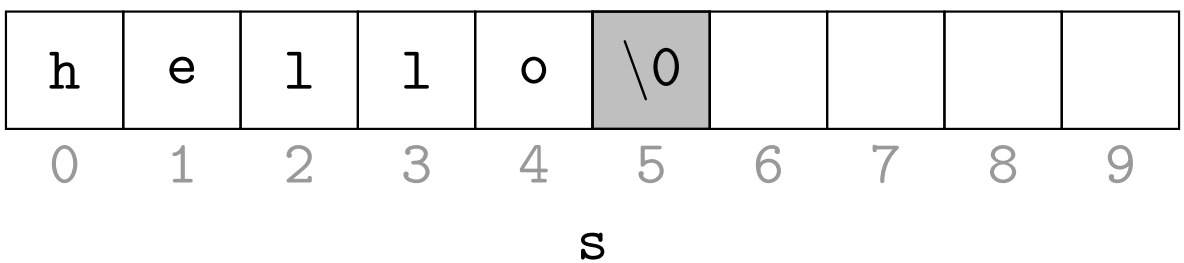
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```



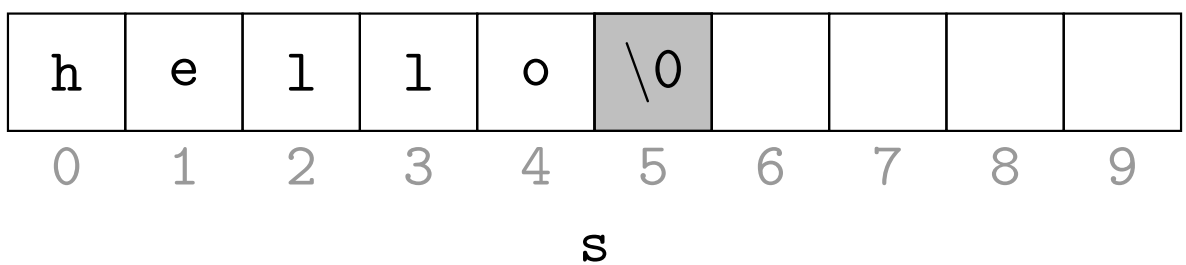
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```



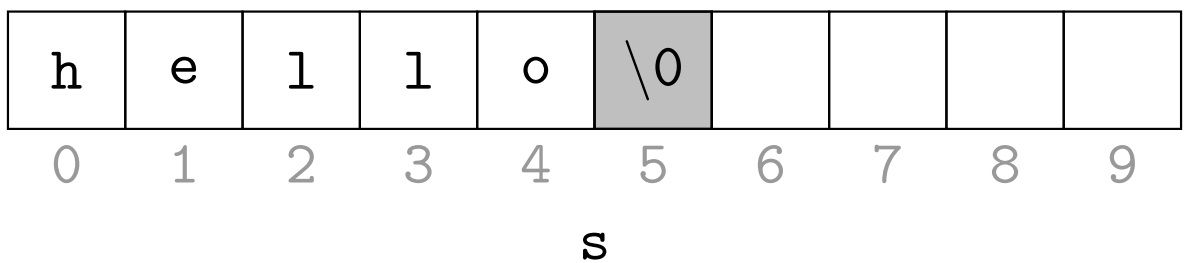
String Length

```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```



String Length

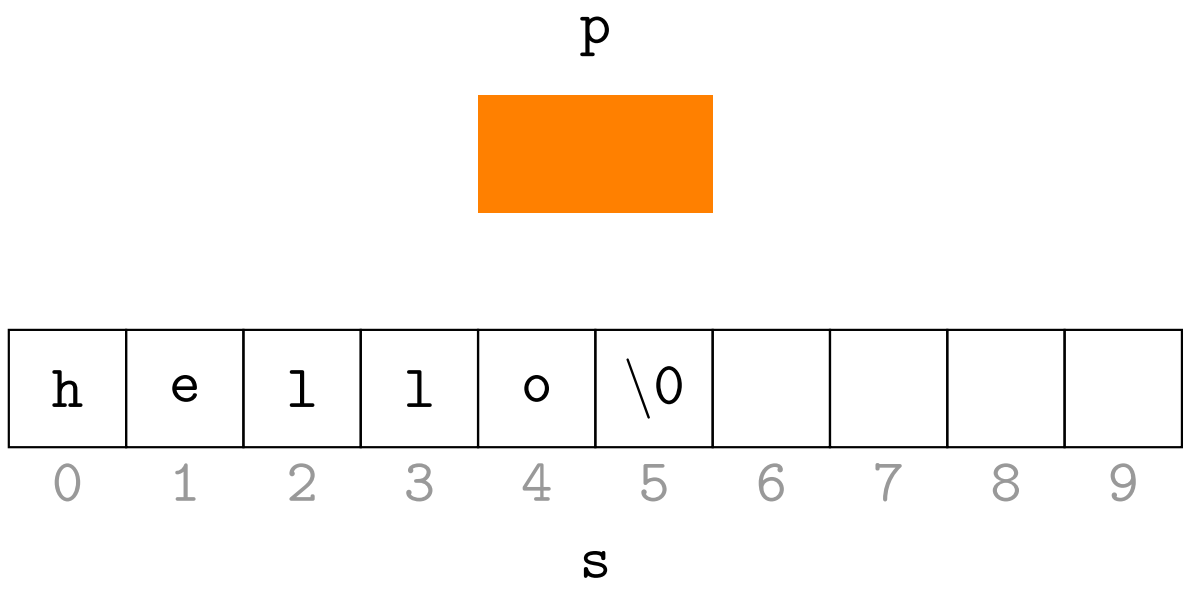
```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```



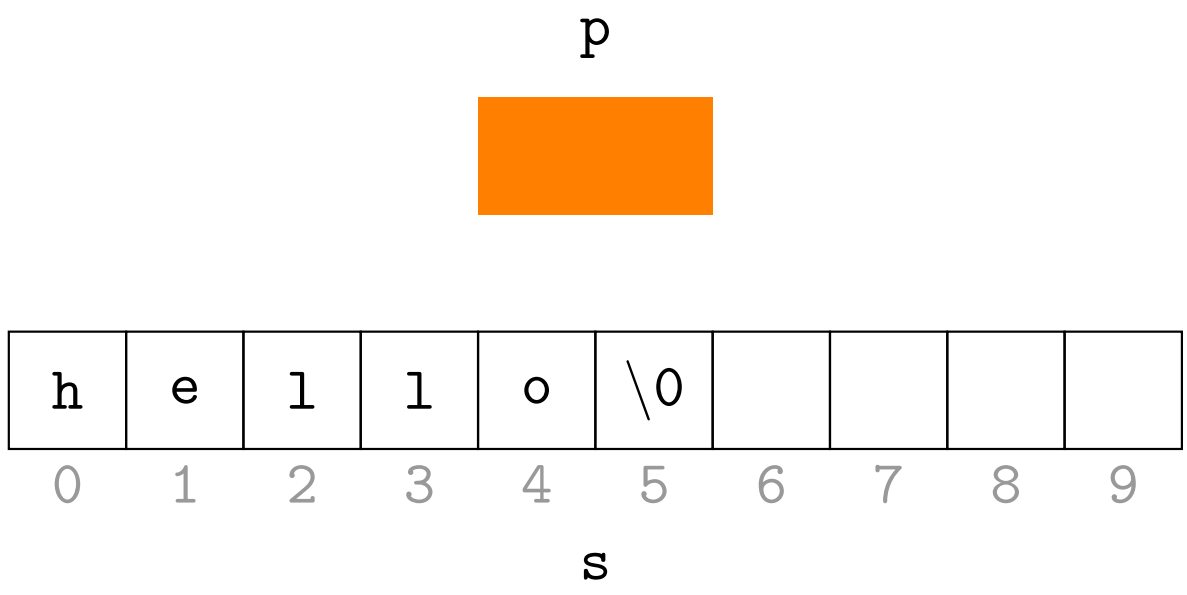
Done!

```
char s[10] = "hello";
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```

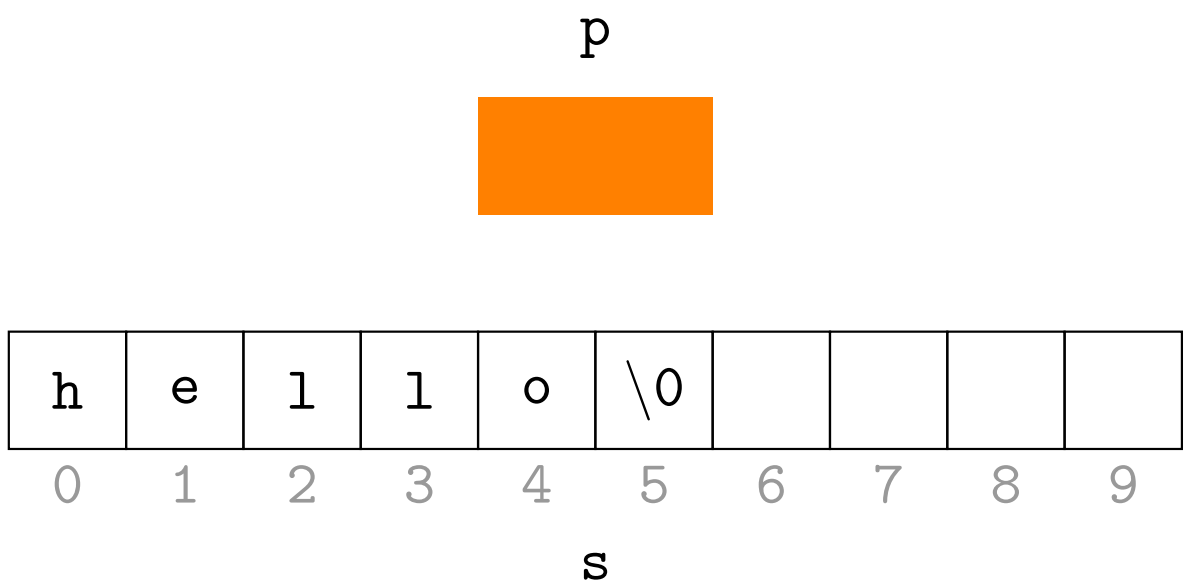
```
char s[10] = "hello";
char *p;
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```



```
char s[10] = "hello";
char *p;
p = s;
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```

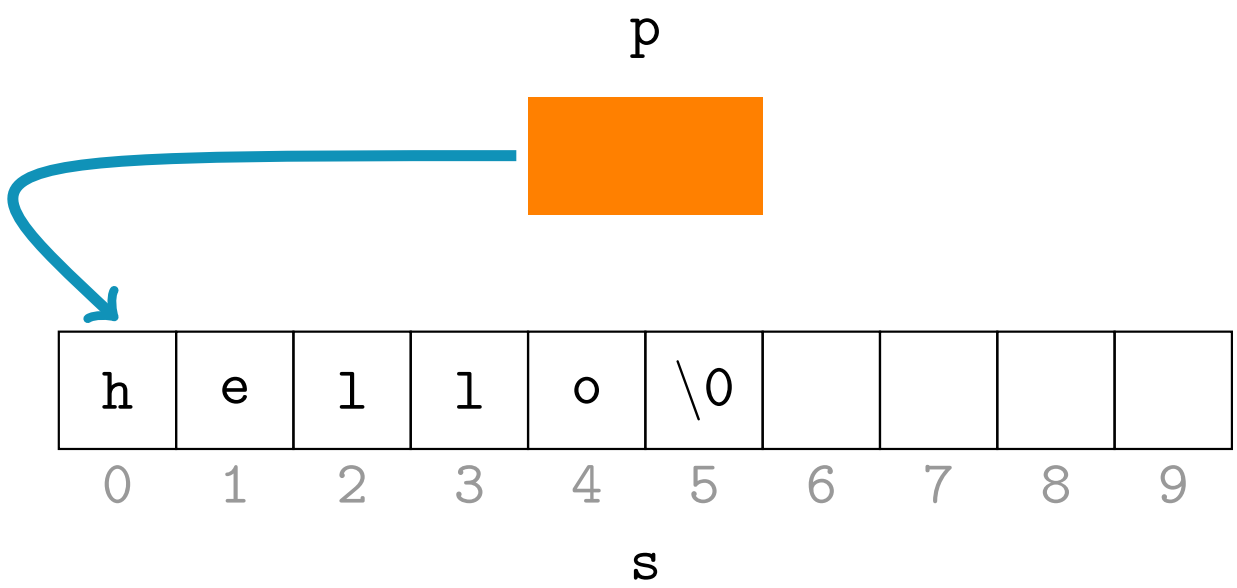


```
char s[10] = "hello";
char *p;
p = s;
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```



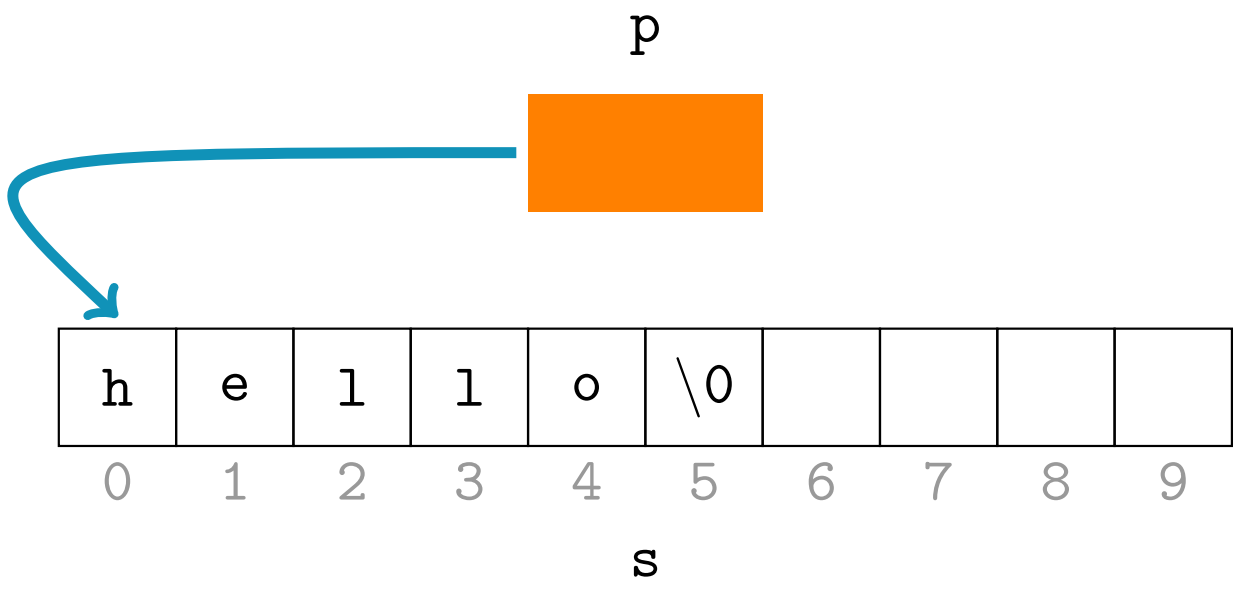
Name of the array stores the address of the first element.

```
char s[10] = "hello";
char *p;
p = s;
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```

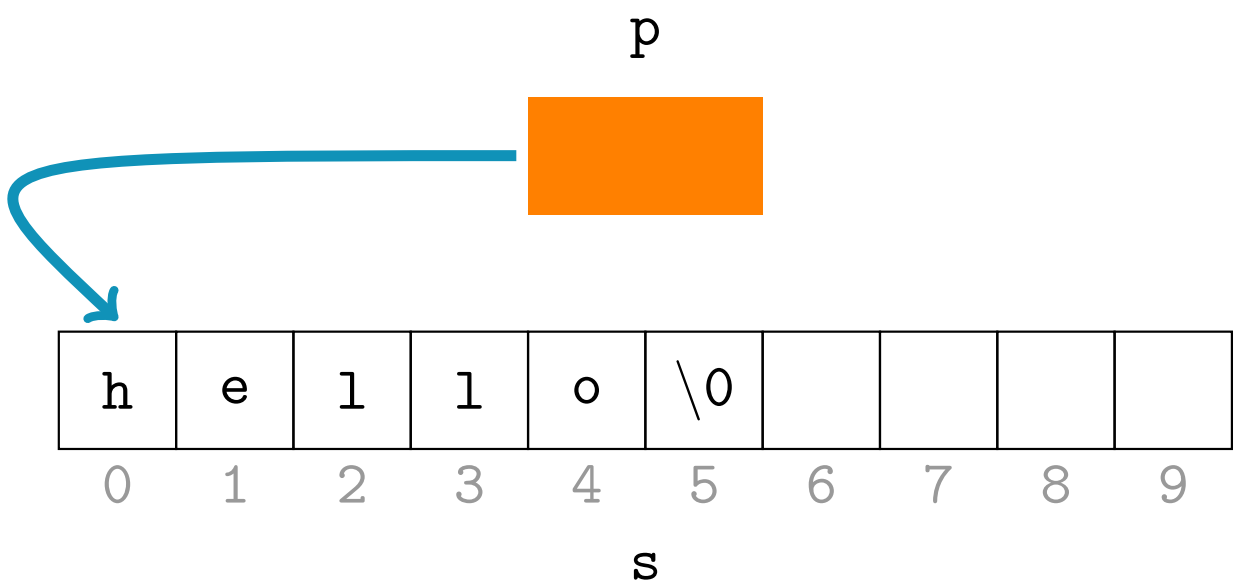


Name of the array stores the address of the first element.

```
char s[10] = "hello";
char *p;
p = s;
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```

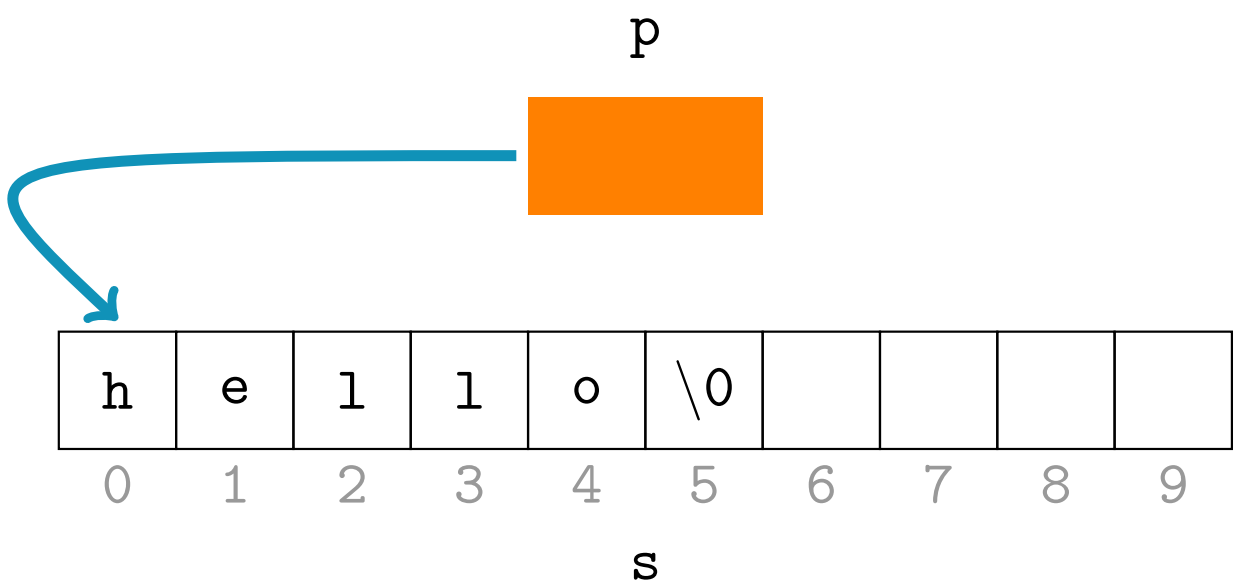


```
char s[10] = "hello";
char *p;
p = s;
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```



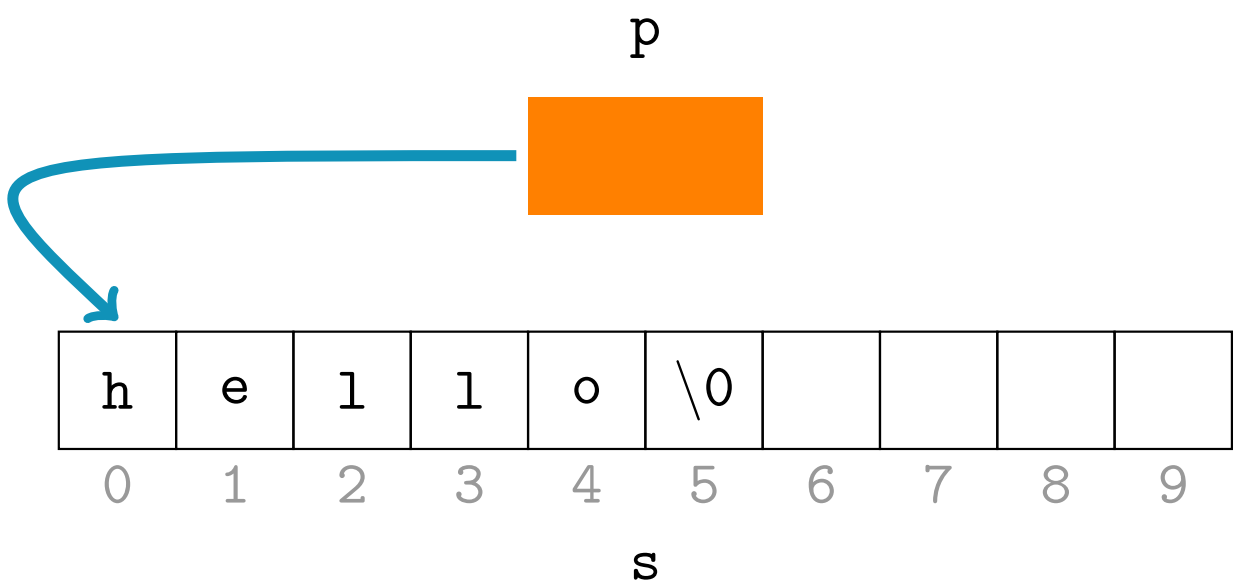
s[i]

```
char s[10] = "hello";
char *p;
p = s;
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```



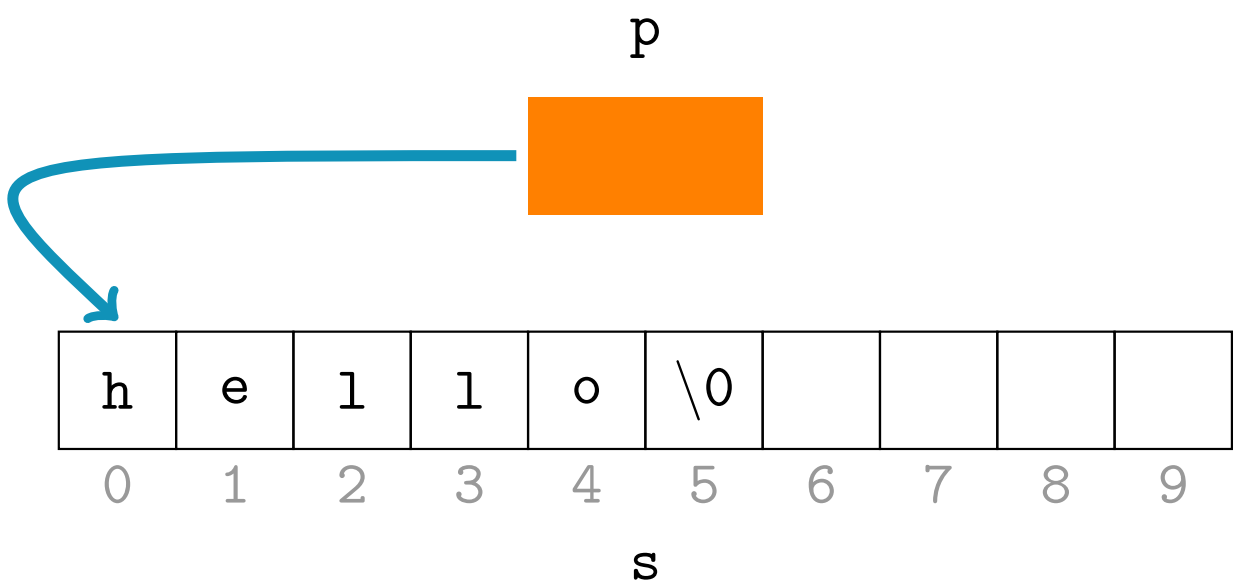
`s[i] = *(s + i)`

```
char s[10] = "hello";
char *p;
p = s;
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```



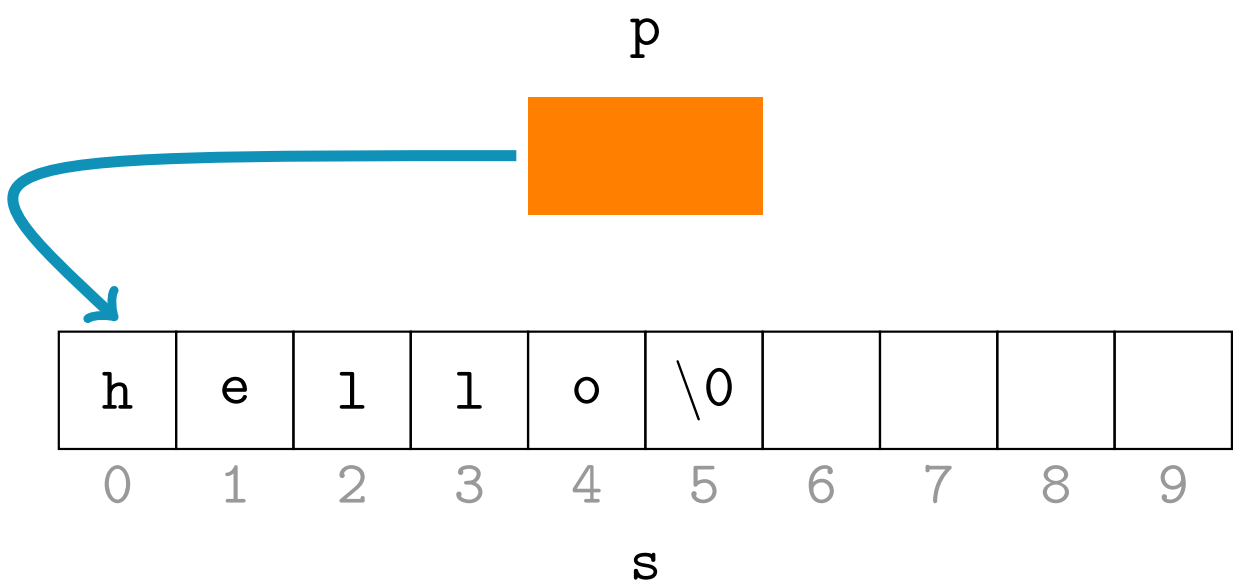
`s[i] = *(s + i) = *(p + i)`

```
char s[10] = "hello";
char *p;
p = s;
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; s[i] != '\0'; i++) {
  nc++;
}
```



`s[i] = *(s + i) = *(p + i) = p[i]`

```
char s[10] = "hello";
char *p;
p = s;
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; p[i] != '\0'; i++) {
  nc++;
}
```



`s[i] = *(s + i) = *(p + i) = p[i]`

```
char s[10] = "hello";
char *p;
p = s;
int nc = 0; /* number of characters */
int i; /* index variable */
for (i = 0; p[i] != '\0'; i++) {
  nc++;
}
```

Comment

If p knows the address of the array, it can calculate its length.

```
int strlen(char *p) {
  char *p;
  p = s;
  int nc = 0; /* number of characters */
  int i; /* index variable */
  for (i = 0; p[i] != '\0'; i++) {
    nc++;
  }
}
```

Comment

If p knows the address of the array, it can calculate its length.

```
int strlen(char *p) {
  char *p;
  p = s;
  int nc = 0; /* number of characters */
  int i; /* index variable */
  for (i = 0; p[i] != '\0'; i++) {
    nc++;
  }
}
```

Comment

If p knows the address of the array, it can calculate its length.

```
int strlen(char *p) {
  char *p;
  p = s;
  int nc = 0; /* number of characters */
  int i; /* index variable */
  for (i = 0; p[i] != '\0'; i++) {
    nc++;
  }
}
```

Comment

If p knows the address of the array, it can calculate its length.

strlen – Version 1

```
int strlen(char *p) {
  int nc = 0; /* number of characters */
  int i; /* index variable */
  for (i = 0; p[i] != '\0'; i++) {
    nc++;
  }
}
```

strlen – Version 1

```
int strlen(char *p) {
  int nc = 0; /* number of characters */
  int i; /* index variable */
  for (i = 0; p[i] != '\0'; i++) {
    nc++;
  }
}
```

"Hello, world!"

strlen – Version 1

```
int strlen(char *p) {
  int nc = 0; /* number of characters */
  int i; /* index variable */
  for (i = 0; p[i] != '\0'; i++) {
    nc++;
  }
}
```

```
strlen("Hello, world!")
```

strlen – Version 1

```
int strlen(char *p) {
  int nc = 0; /* number of characters */
  int i; /* index variable */
  for (i = 0; p[i] != '\0'; i++) {
    nc++;
  }
}

printf("%d", strlen("Hello, world!"));
```

strlen – Version 1

```
int strlen(char *p) {
  int nc = 0; /* number of characters */
  int i; /* index variable */
  for (i = 0; p[i] != '\0'; i++) {
    nc++;
  }
}

printf("%d", strlen("Hello, world!")); → 13
```

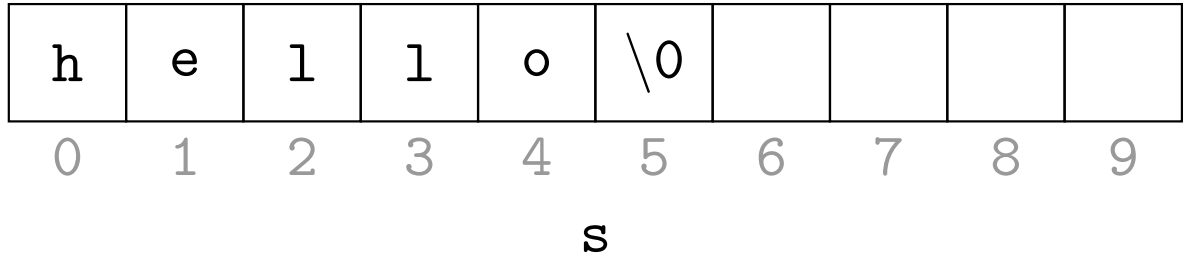
String Length – Version 2

String Length – Version 2

```
int strlen(char *p) {




}
```

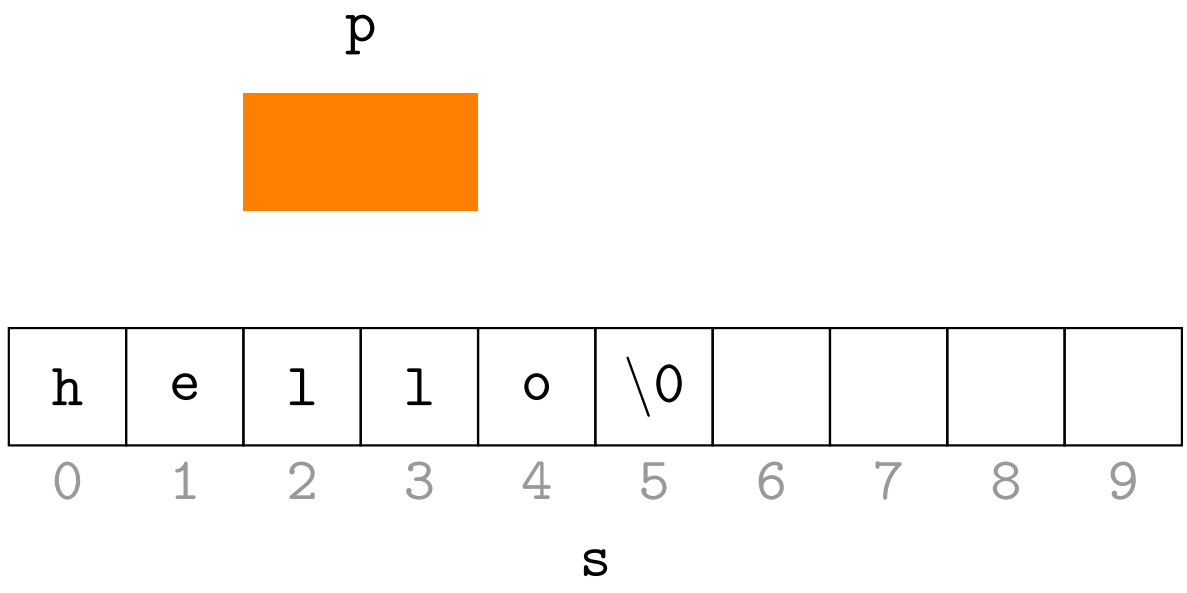
String Length – Version 2

```
int strlen(char *p) {




}
```



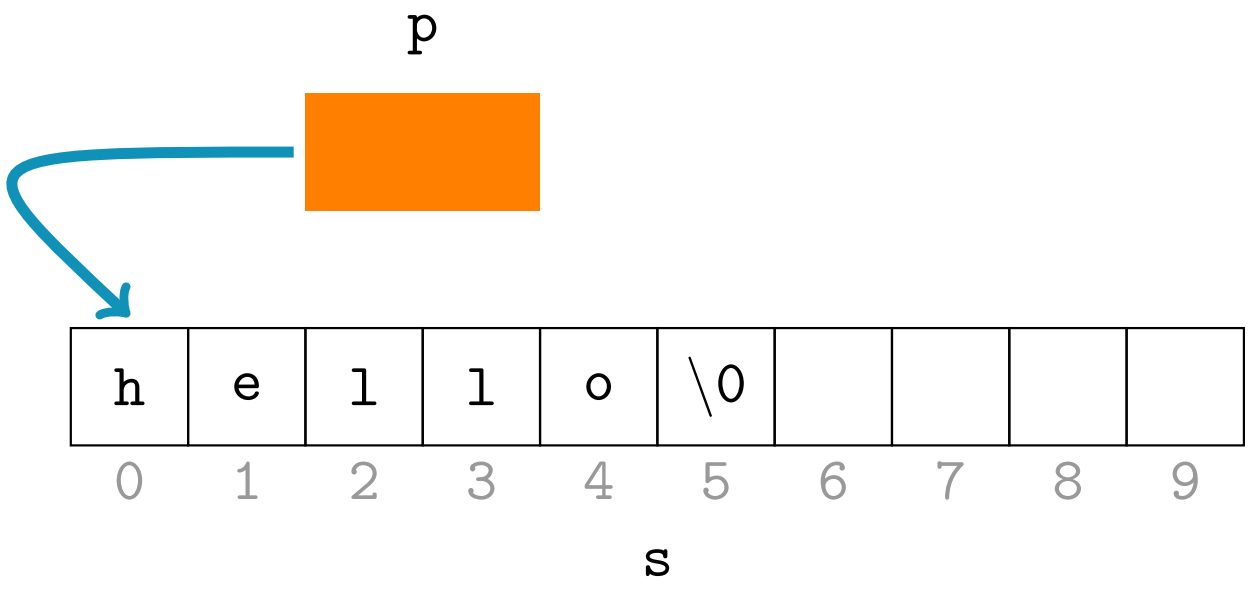
String Length – Version 2

```
int strlen(char *p) {




}
```



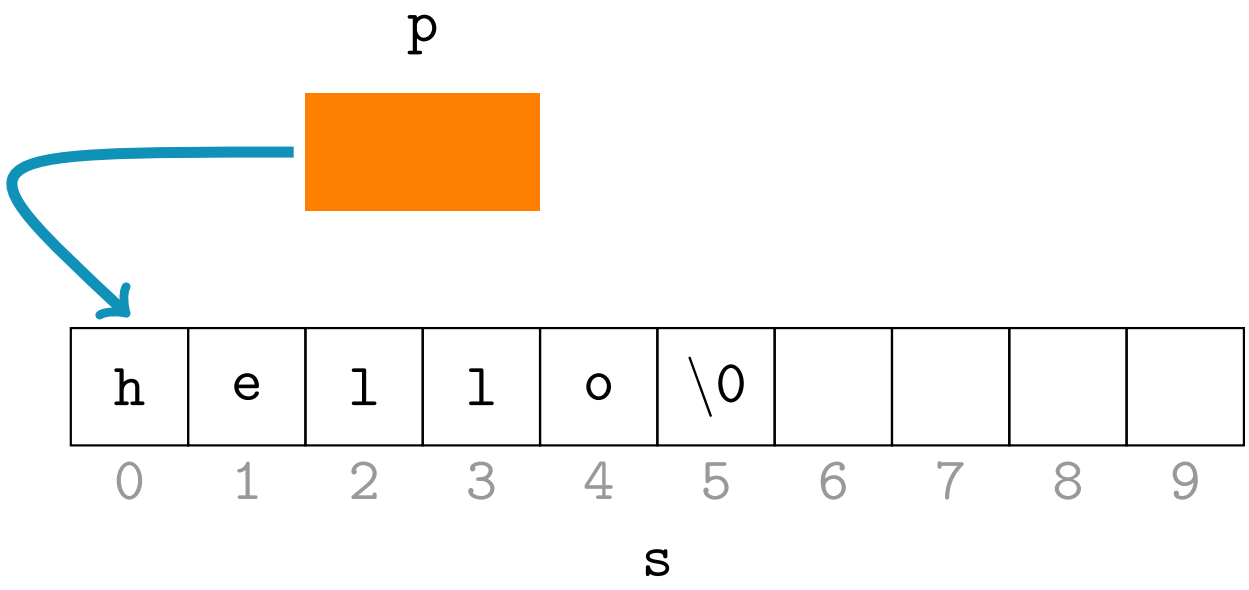
String Length – Version 2

```
int strlen(char *p) {




}
```



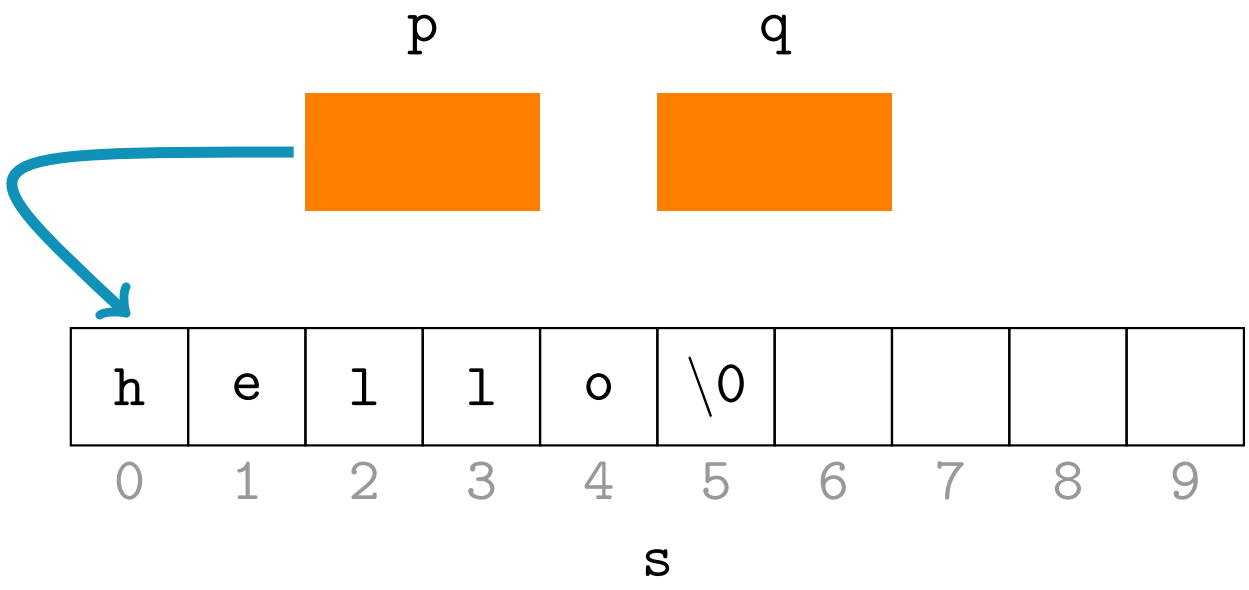
String Length – Version 2

```
int strlen(char *p) {
  char *q;




}
```



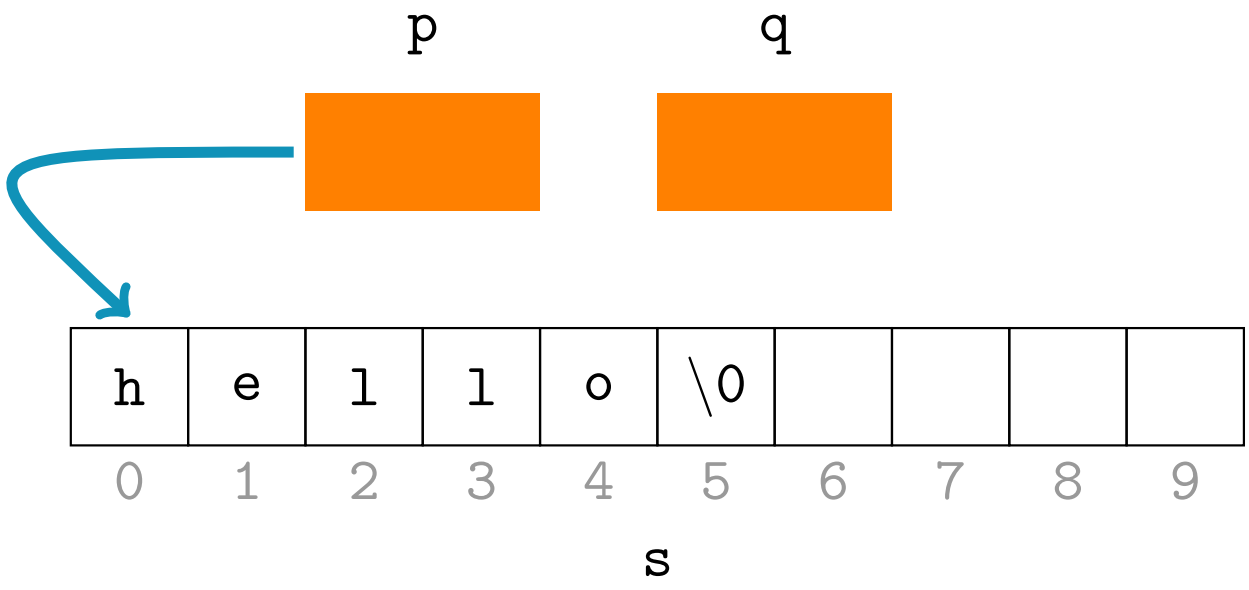
String Length – Version 2

```
int strlen(char *p) {
  char *q;




}
```



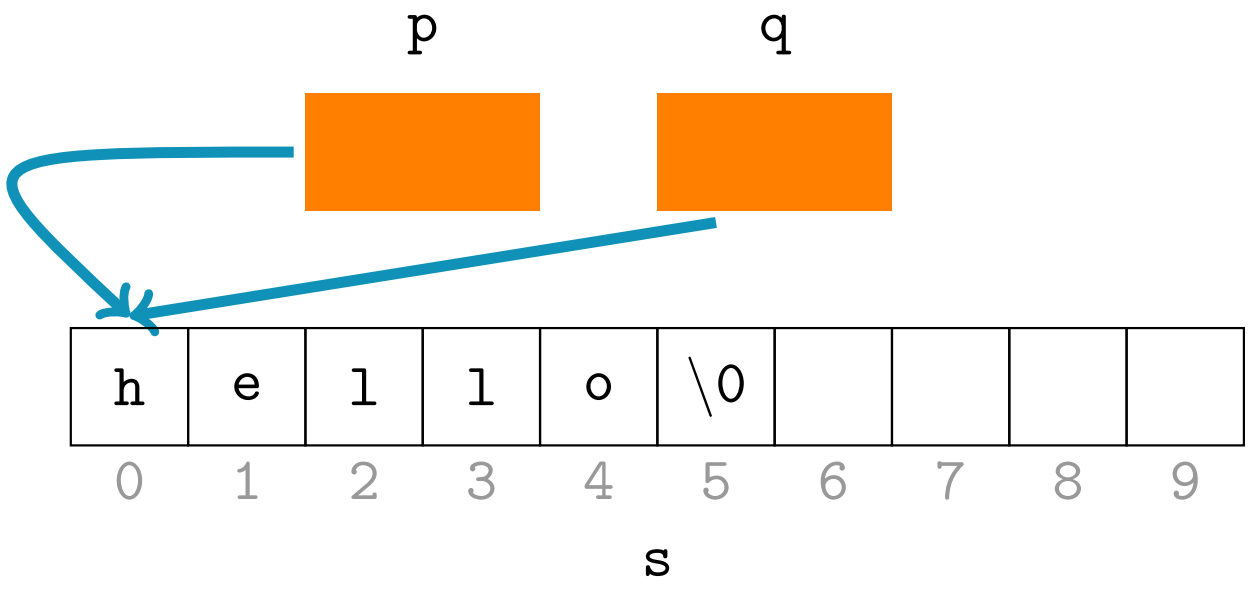
String Length – Version 2

```
int strlen(char *p) {
  char *q;
  q = p;



}
```



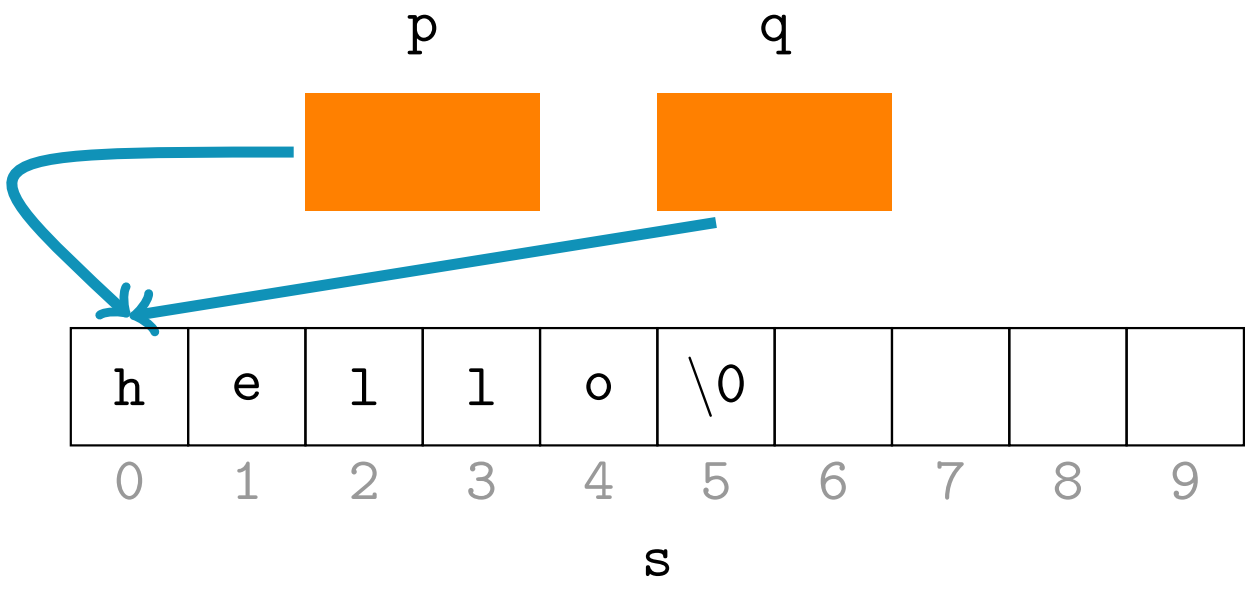
String Length – Version 2

```
int strlen(char *p) {
  char *q;
  q = p;



}
```



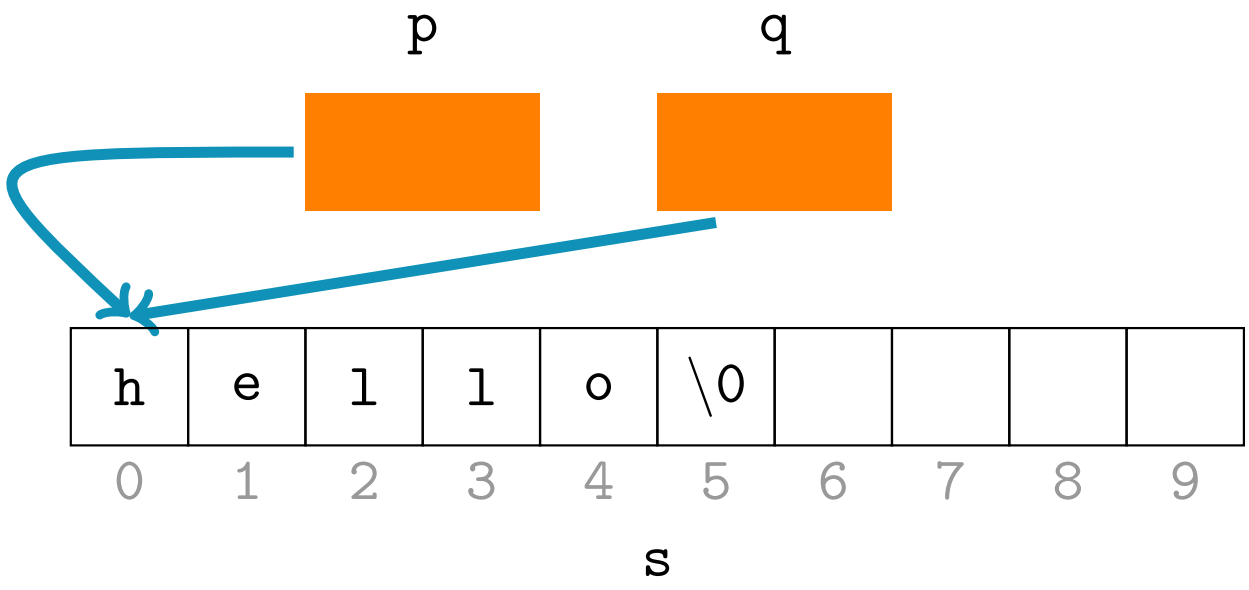
String Length – Version 2

```
int strlen(char *p) {
  char *q;
  q = p;
  while (*q != '\0')
    q++;

}
```



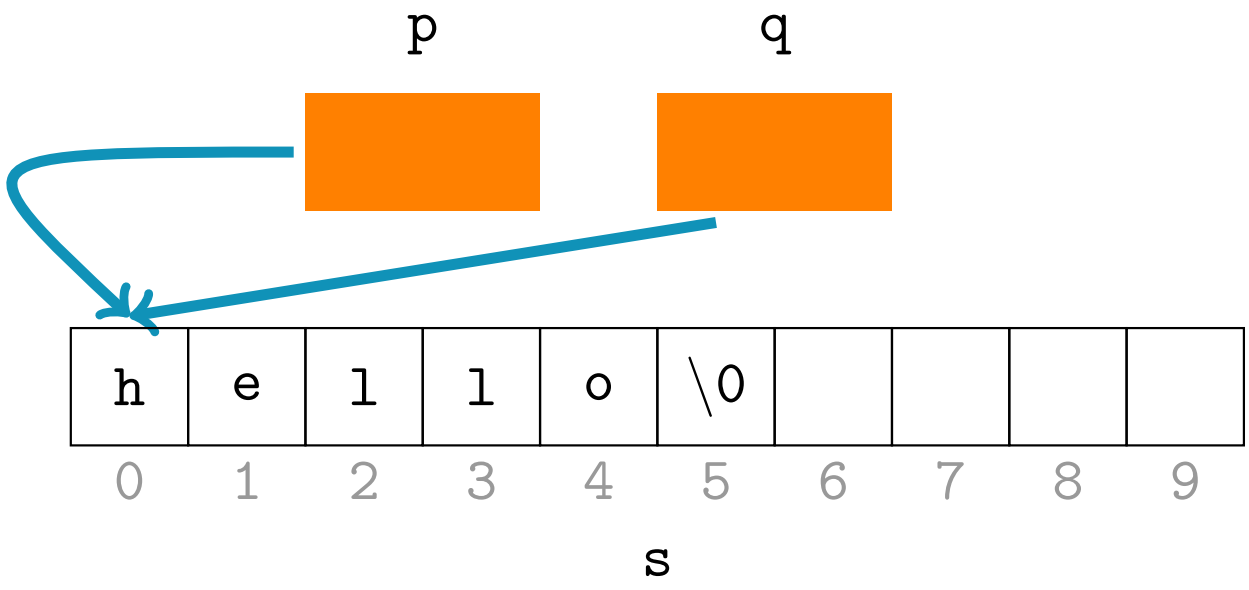
String Length – Version 2

```
int strlen(char *p) {
  char *q;
  q = p;
  while (*q != '\0')
    q++;

}
```



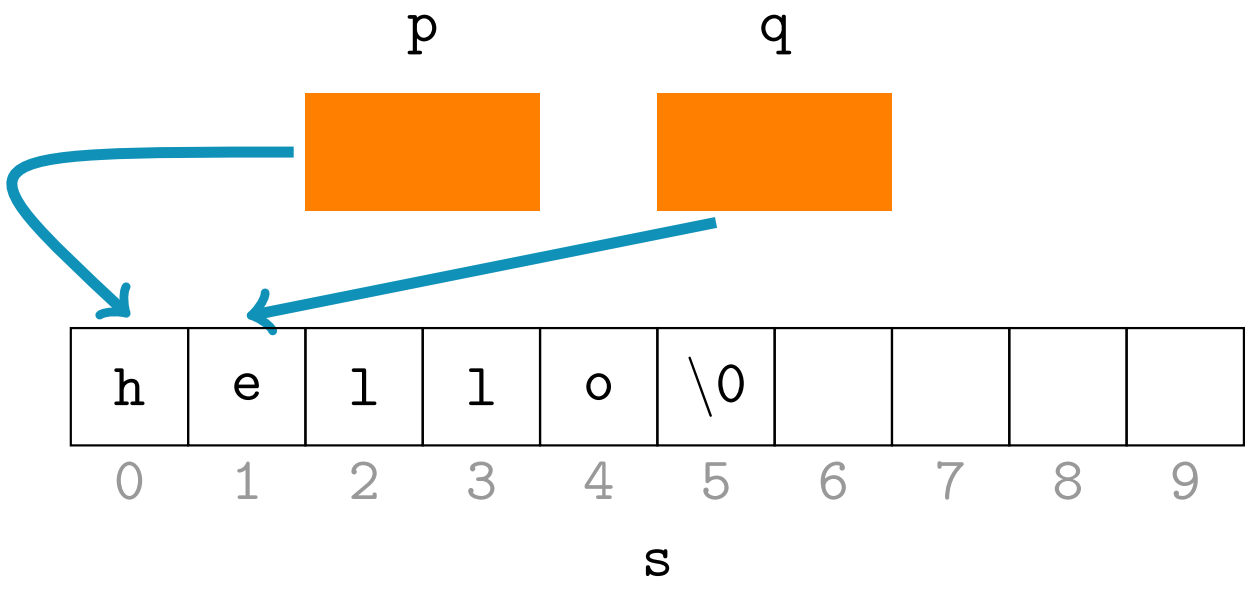
String Length – Version 2

```
int strlen(char *p) {
  char *q;
  q = p;
  while (*q != '\0')
    q++;

}
```



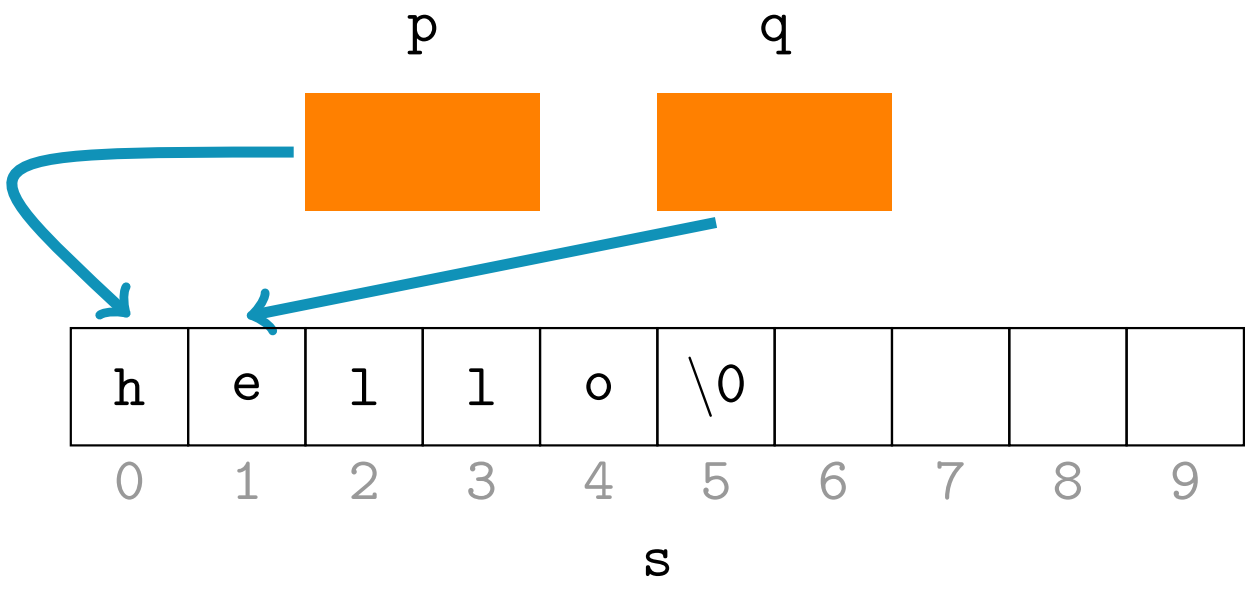
String Length – Version 2

```
int strlen(char *p) {
  char *q;
  q = p;
  while (*q != '\0')
    q++;

}
```



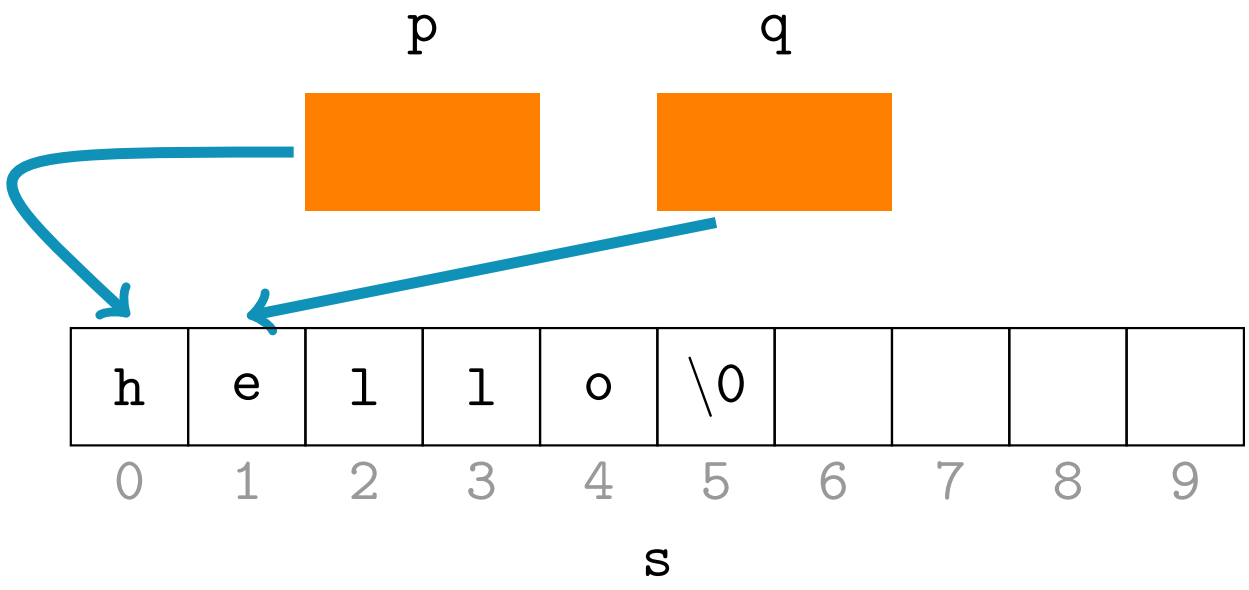
String Length – Version 2

```
int strlen(char *p) {
  char *q;
  q = p;
  while (*q != '\0')
    q++;

}
```



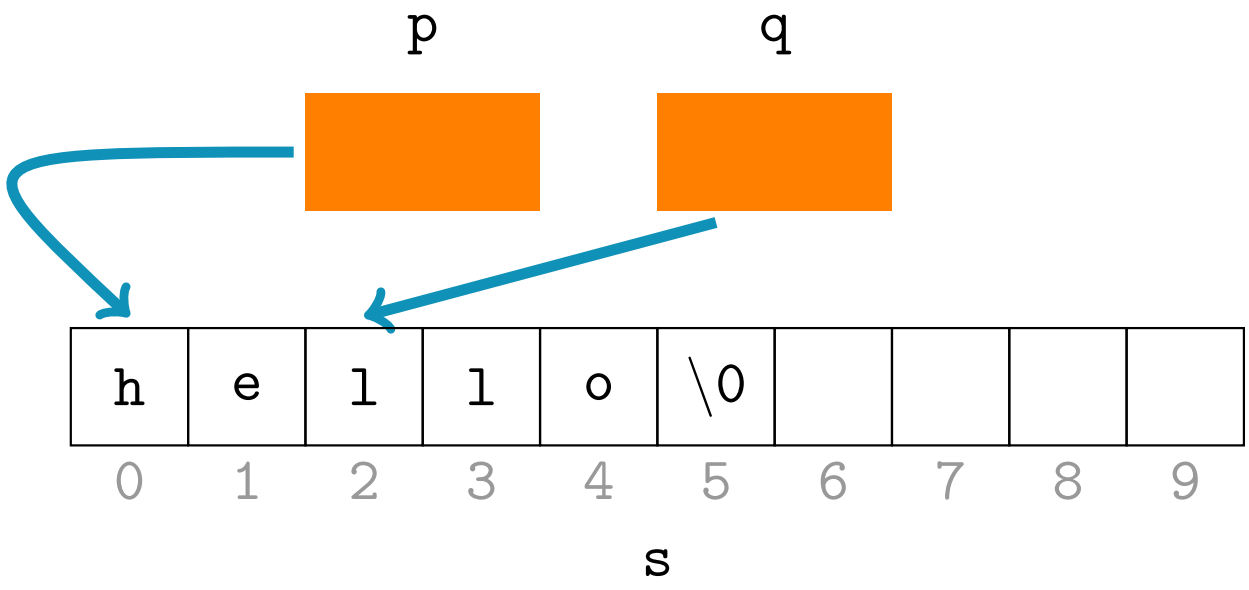
String Length – Version 2

```
int strlen(char *p) {
  char *q;
  q = p;
  while (*q != '\0')
    q++;

}
```



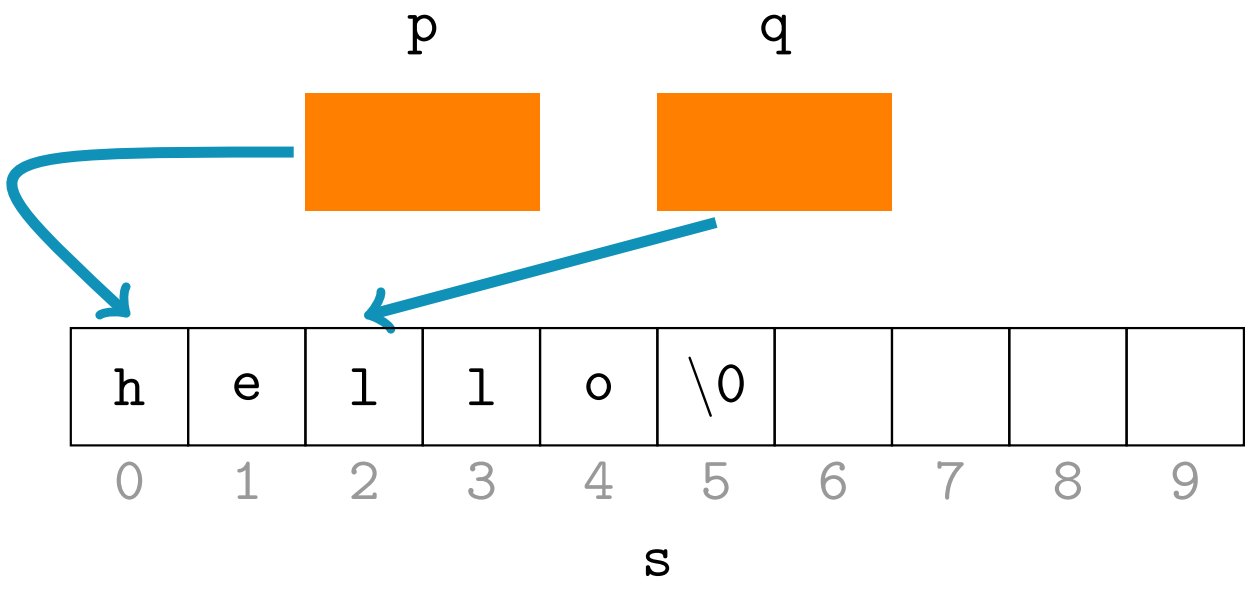
String Length – Version 2

```
int strlen(char *p) {
  char *q;
  q = p;
  while (*q != '\0')
    q++;

}
```



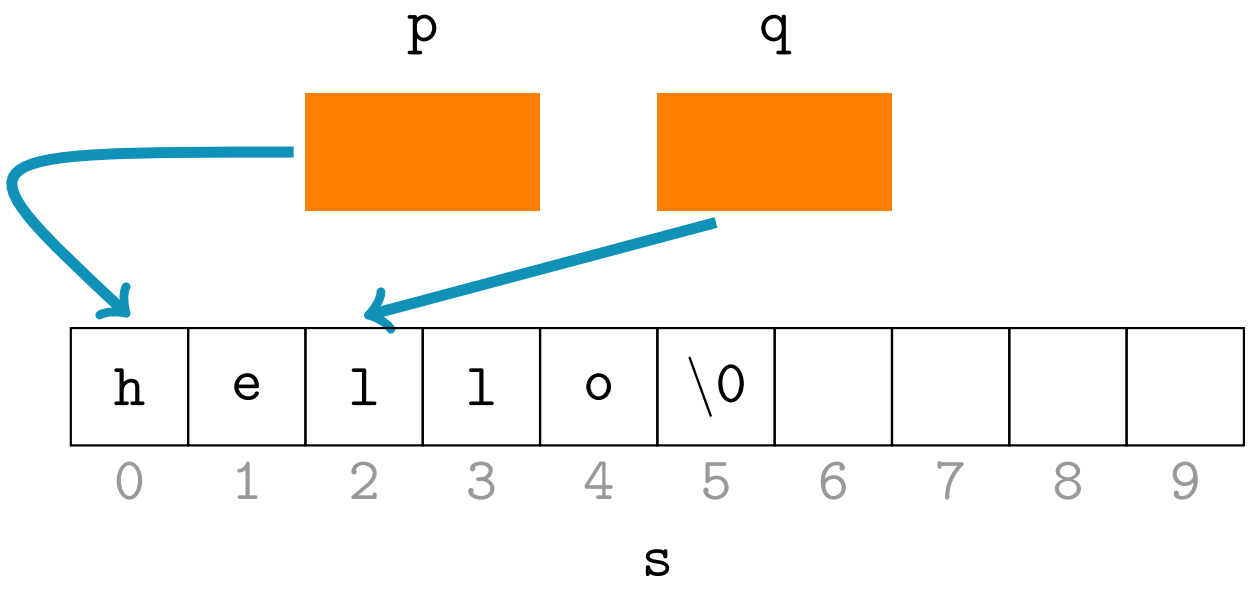
String Length – Version 2

```
int strlen(char *p) {
  char *q;
  q = p;
  while (*q != '\0')
    q++;

}
```



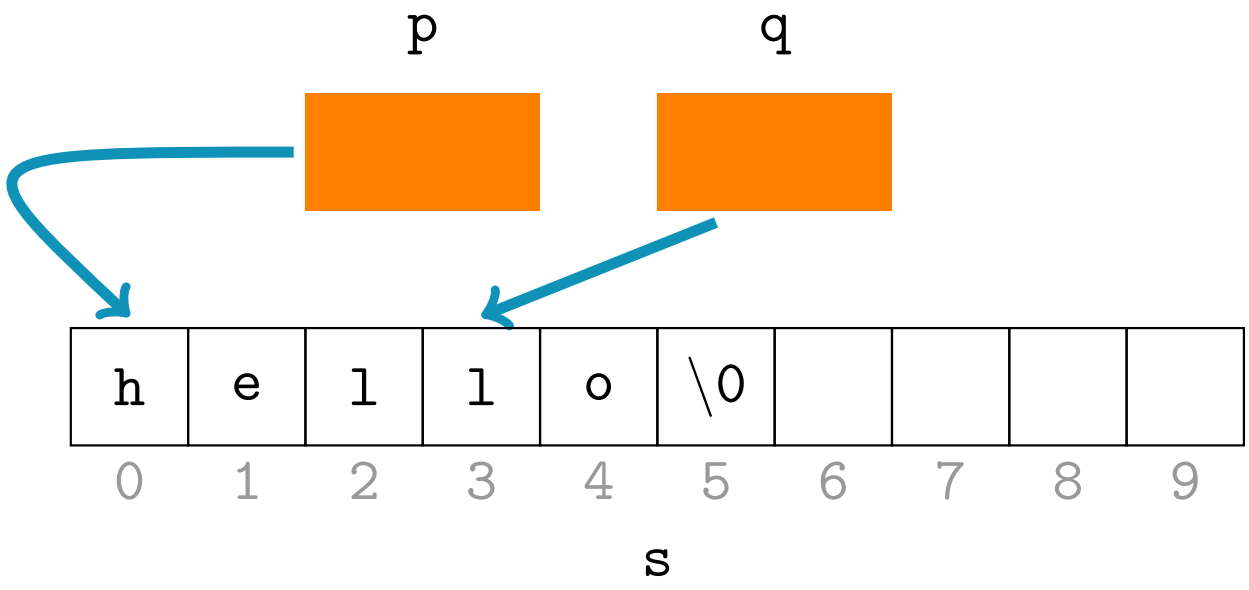
String Length – Version 2

```
int strlen(char *p) {
  char *q;
  q = p;
  while (*q != '\0')
    q++;

}
```



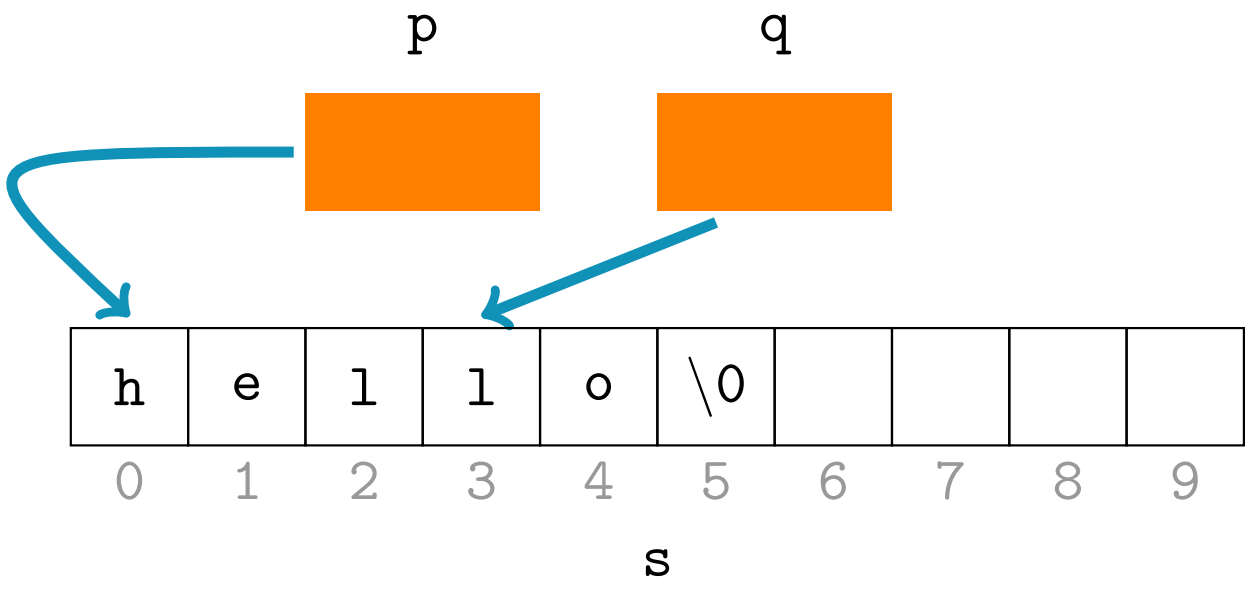
String Length – Version 2

```
int strlen(char *p) {
  char *q;
  q = p;
  while (*q != '\0')
    q++;

}
```



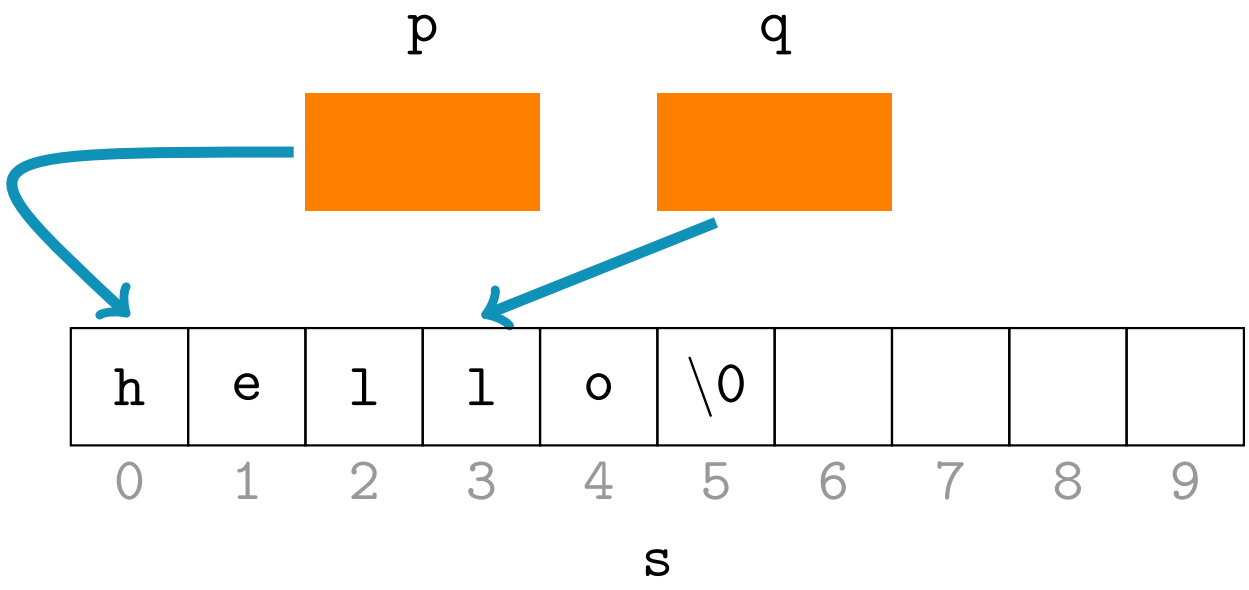
String Length – Version 2

```
int strlen(char *p) {
  char *q;
  q = p;
  while (*q != '\0')
    q++;

}
```



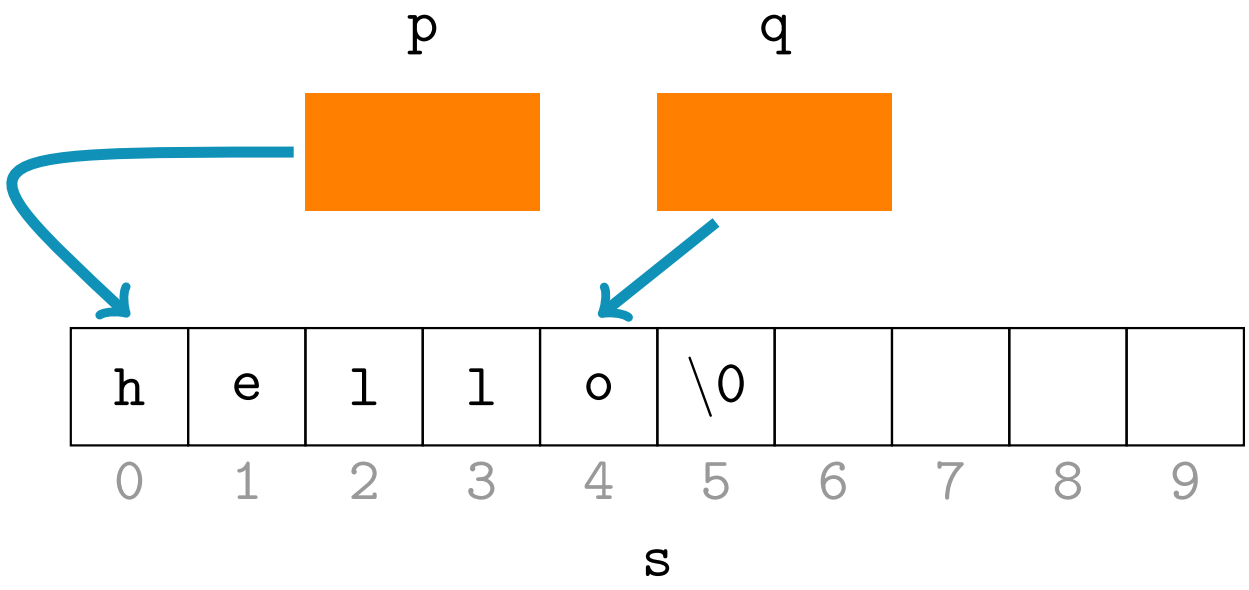
String Length – Version 2

```
int strlen(char *p) {
  char *q;
  q = p;
  while (*q != '\0')
    q++;

}
```



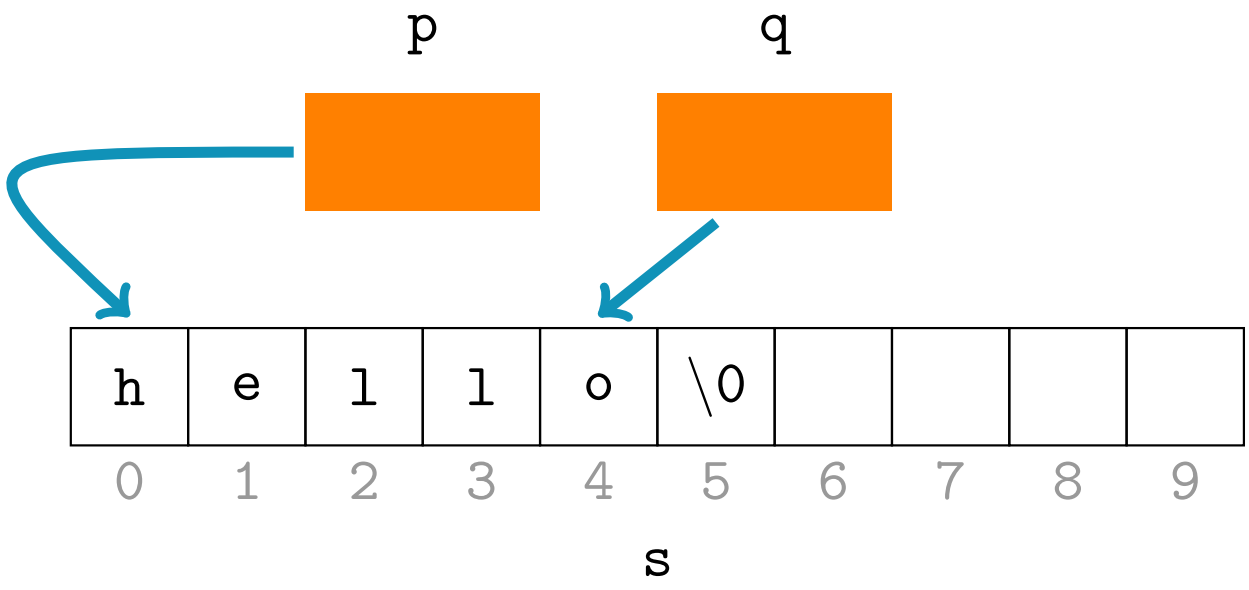
String Length – Version 2

```
int strlen(char *p) {
  char *q;
  q = p;
  while (*q != '\0')
    q++;

}
```



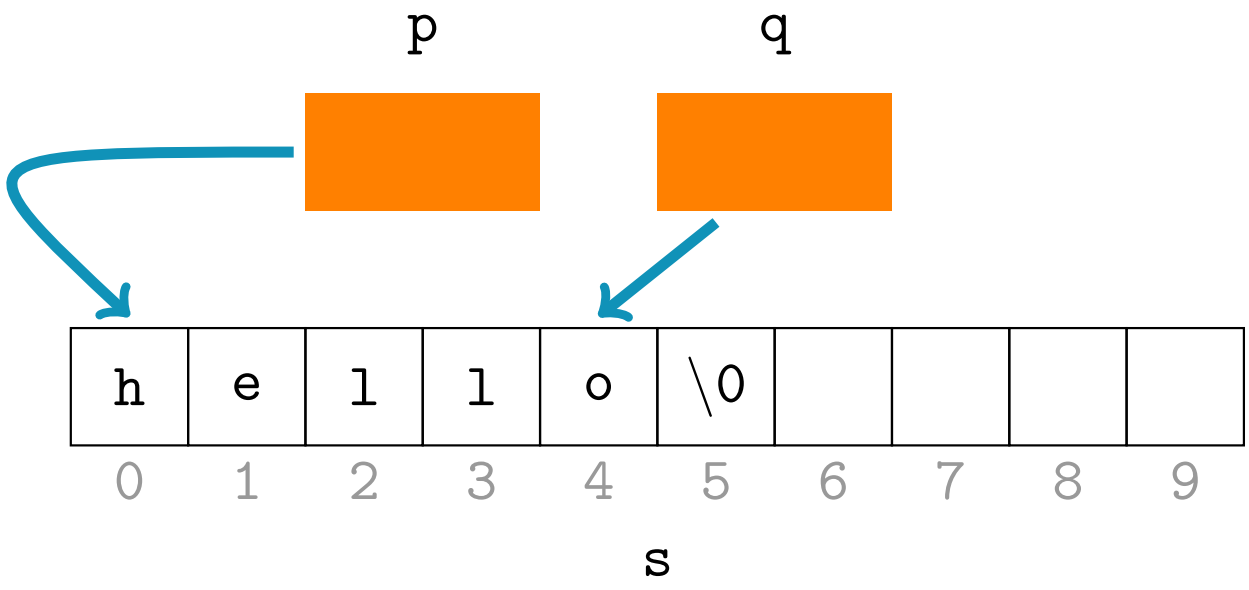
String Length – Version 2

```
int strlen(char *p) {
  char *q;
  q = p;
  while (*q != '\0')
    q++;

}
```



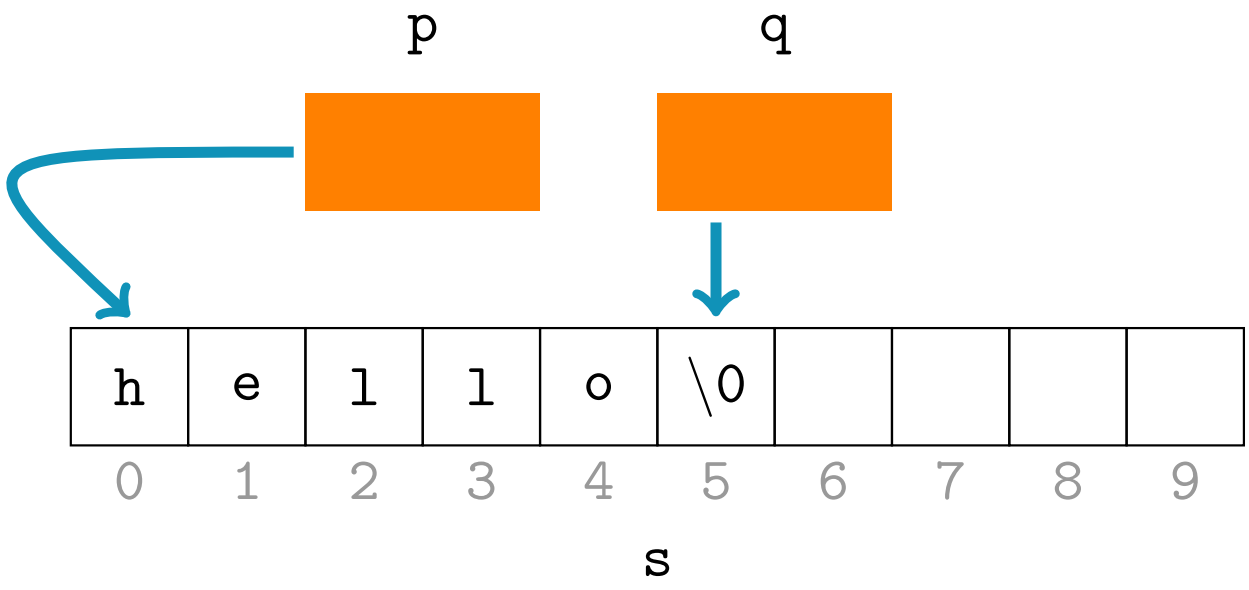
String Length – Version 2

```
int strlen(char *p) {
  char *q;
  q = p;
  while (*q != '\0')
    q++;

}
```



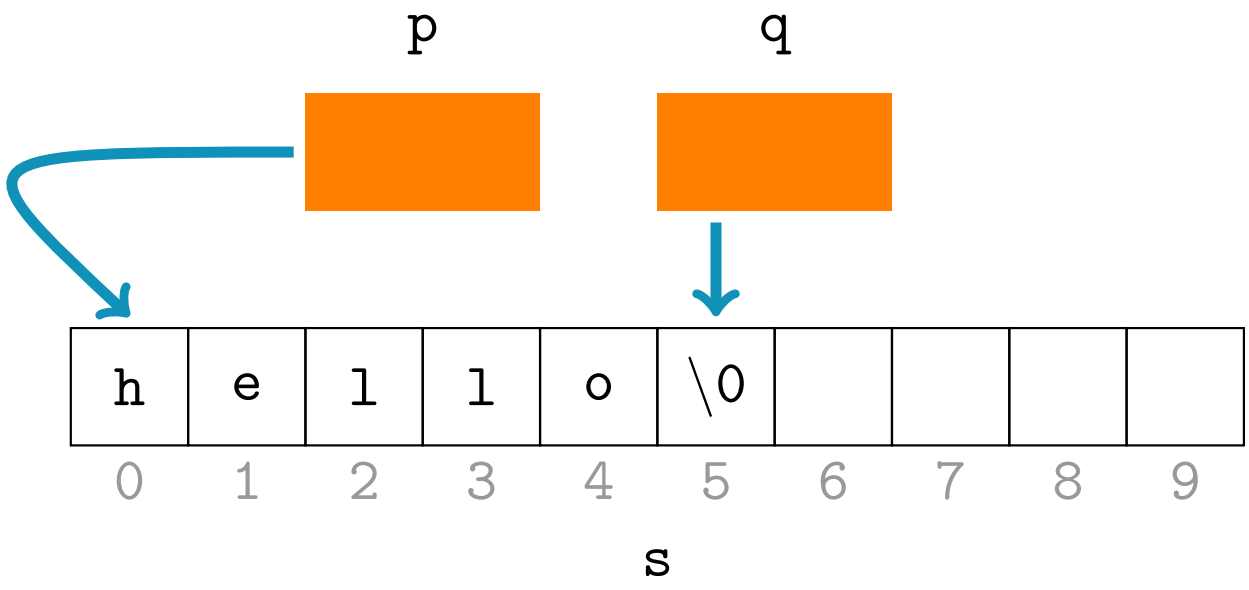
String Length – Version 2

```
int strlen(char *p) {
  char *q;
  q = p;
  while (*q != '\0')
    q++;

}
```



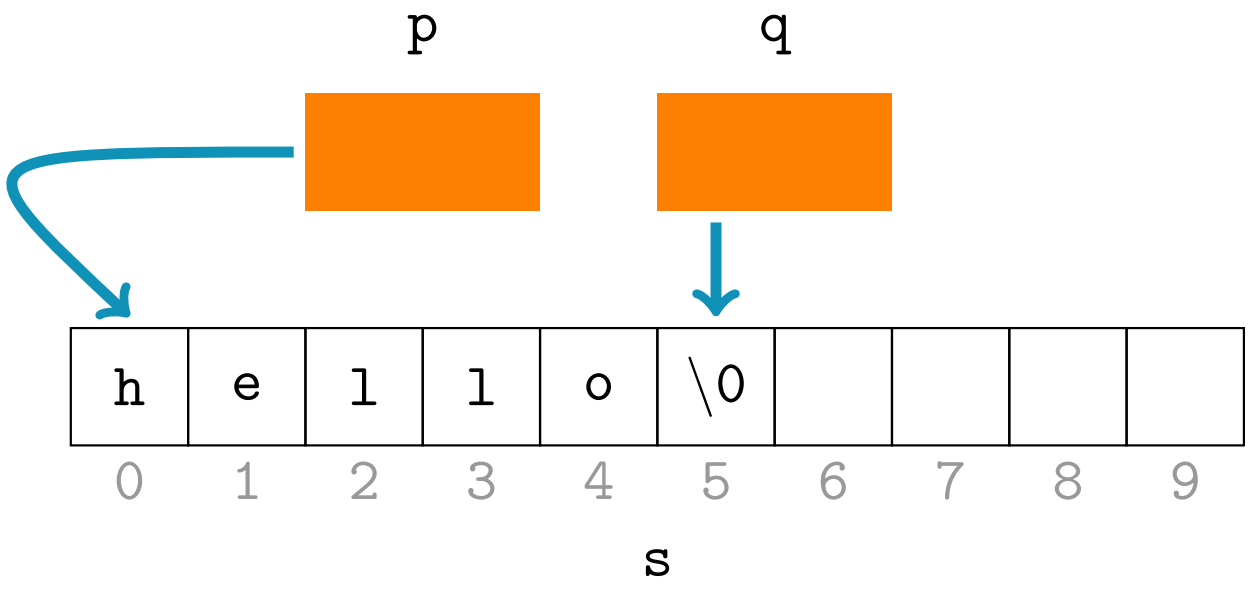
String Length – Version 2

```
int strlen(char *p) {
  char *q;
  q = p;
  while (*q != '\0')
    q++;

}
```



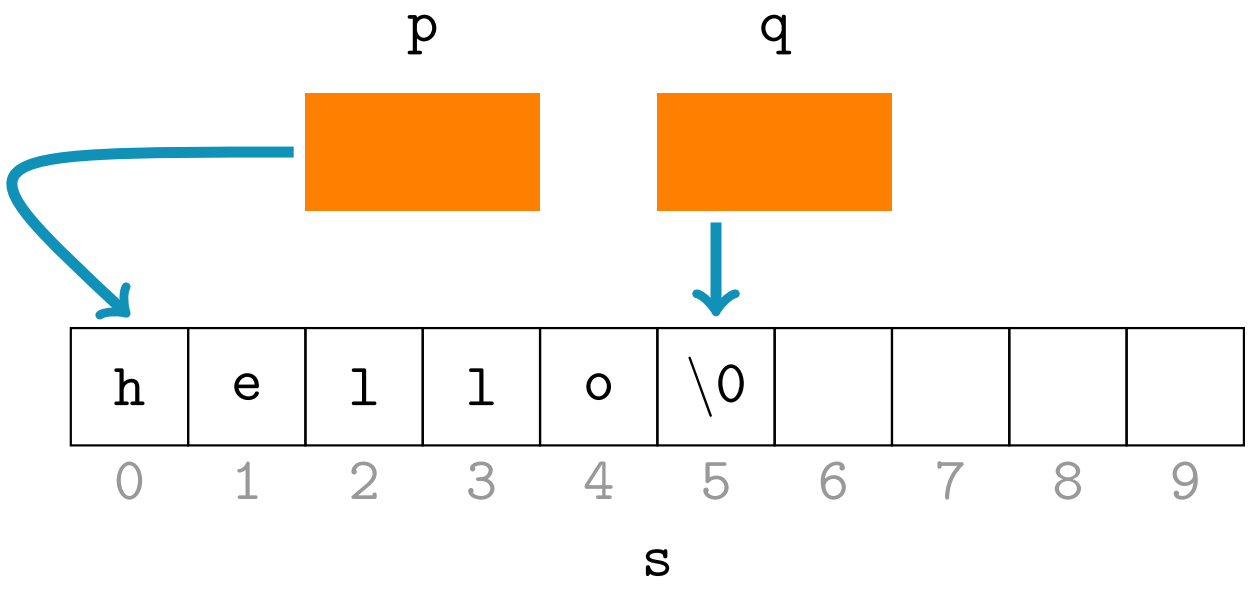
String Length – Version 2

```
int strlen(char *p) {
  char *q;
  q = p;
  while (*q != '\0')
    q++;

}
```



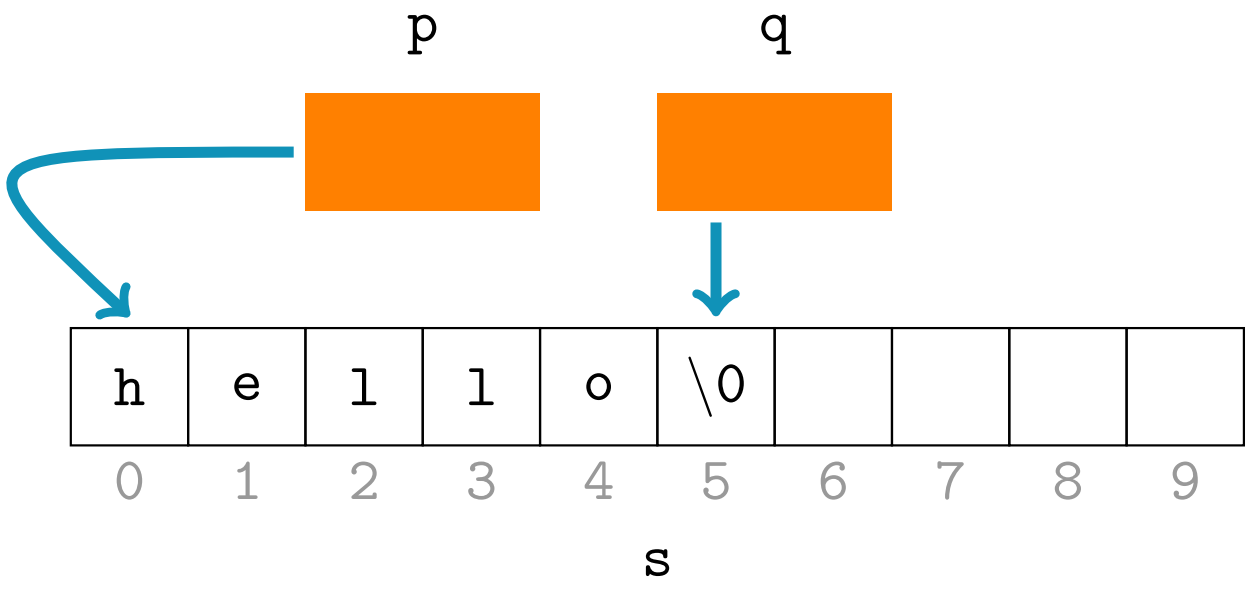
String Length – Version 2

```
int strlen(char *p) {
  char *q;
  q = p;
  while (*q != '\0')
    q++;
        q - p
}
```



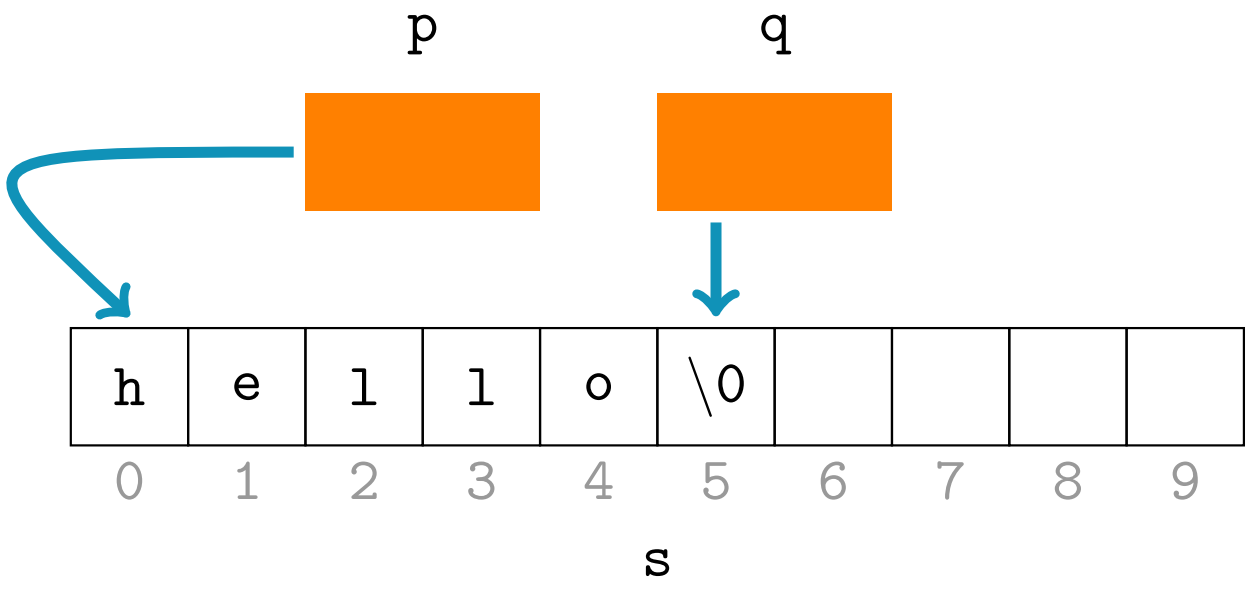
String Length – Version 2

```
int strlen(char *p) {
  char *q;
  q = p;
  while (*q != '\0')
    q++;
        q - p  /* length of the string */
}
```



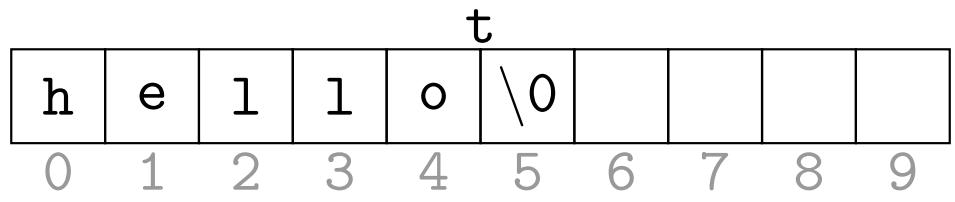
String Length – Version 2

```
int strlen(char *p) {
  char *q;
  q = p;
  while (*q != '\0')
    q++;
  return q - p; /* length of the string */
}
```

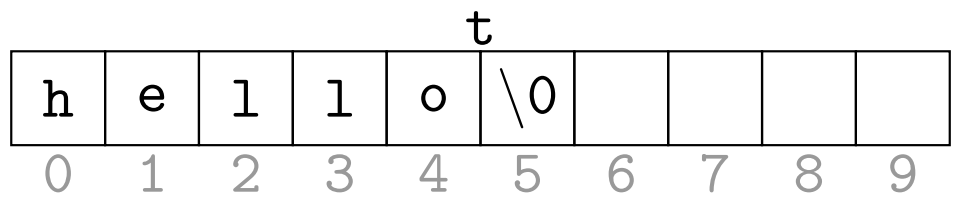


```
char t[10]="hello";
```

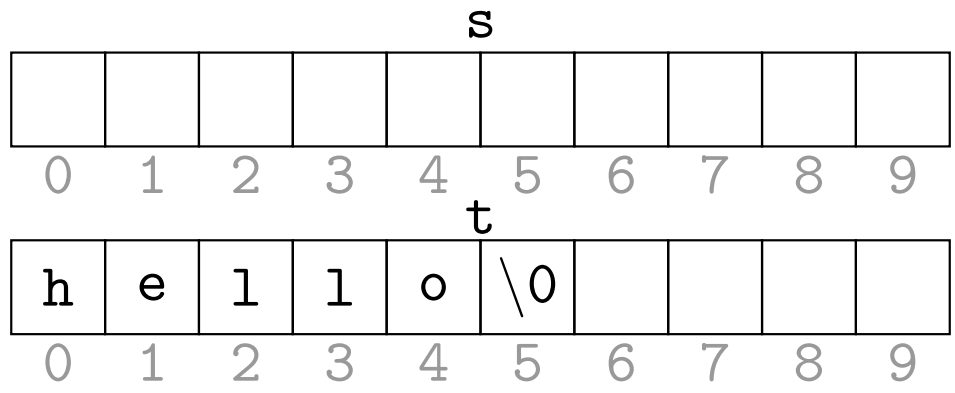
```
char t[10]="hello";
```



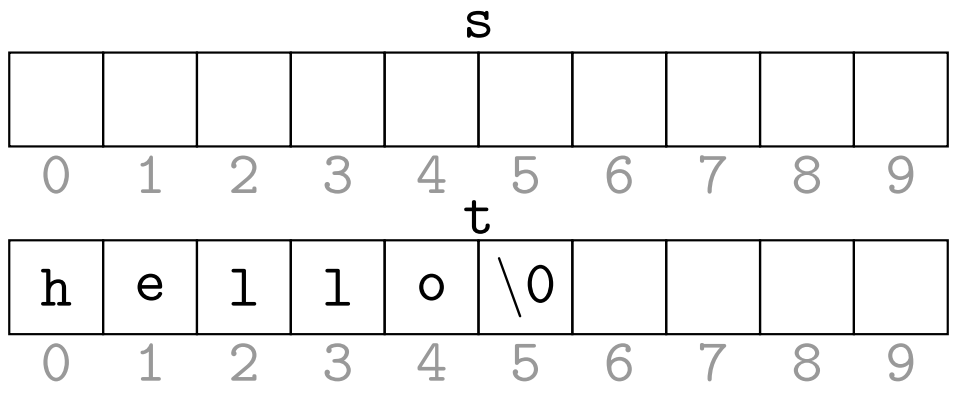
```
char t[10]="hello", s[10];
```



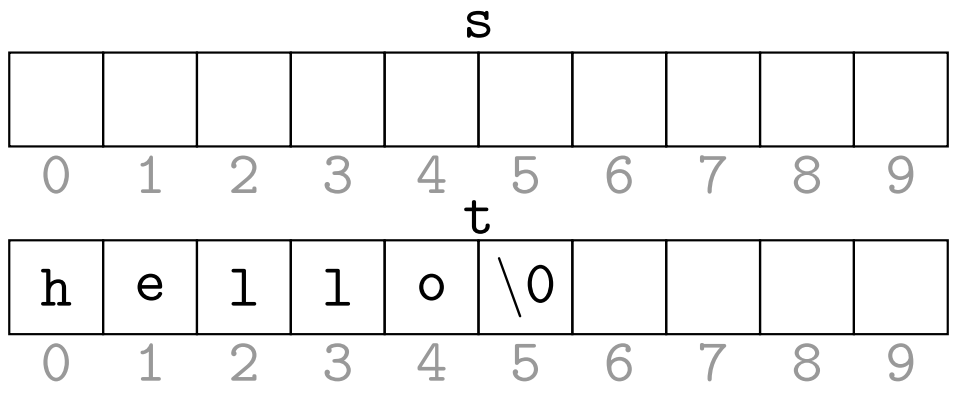
```
char t[10]="hello", s[10];
```



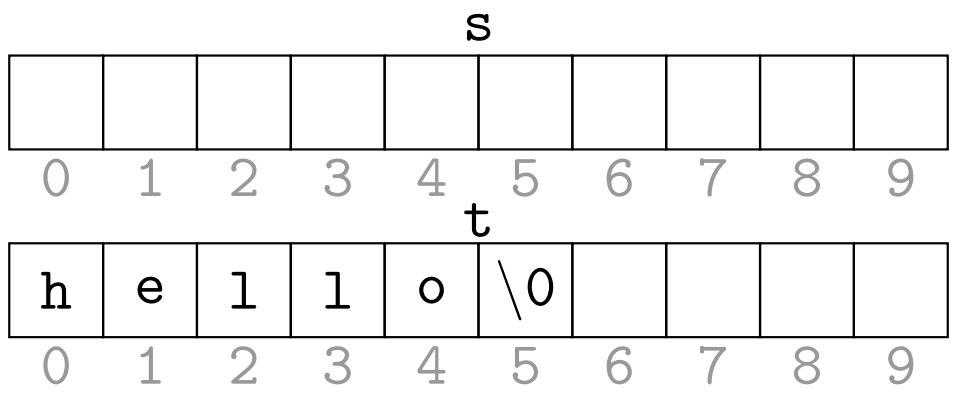
```
char t[10]="hello", s[10];/*copy t to s*/
```



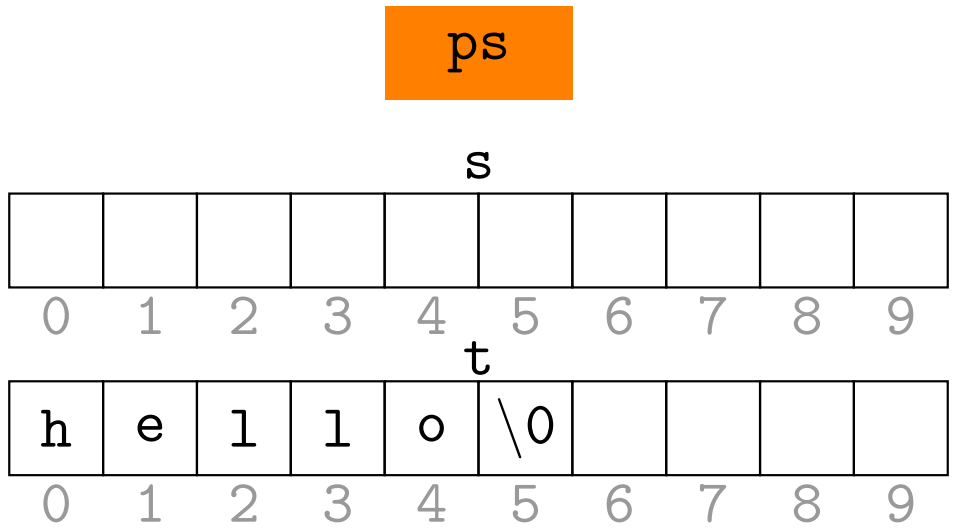
```
char t[10]="hello", s[10];/*copy t to s - pointers*/
```



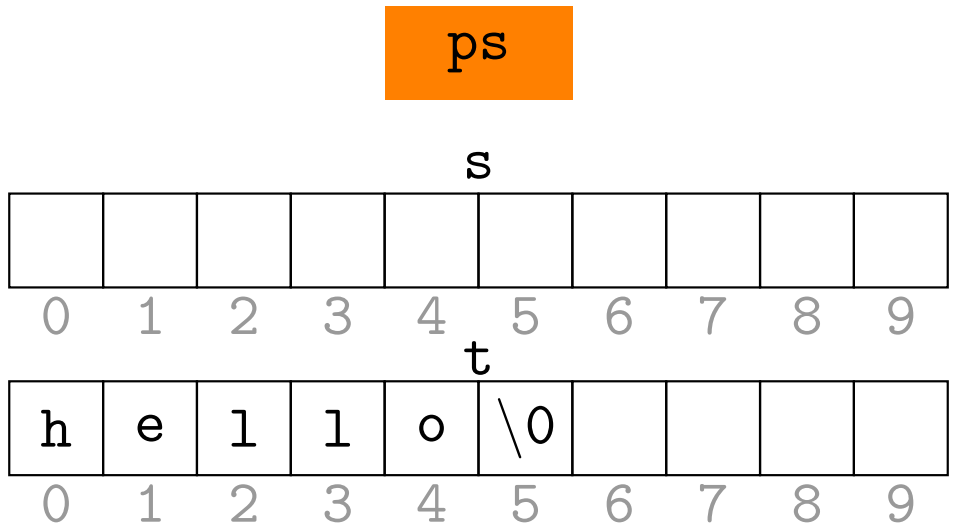
```
char t[10]="hello", s[10];/*copy t to s - pointers*/
char *ps;
```



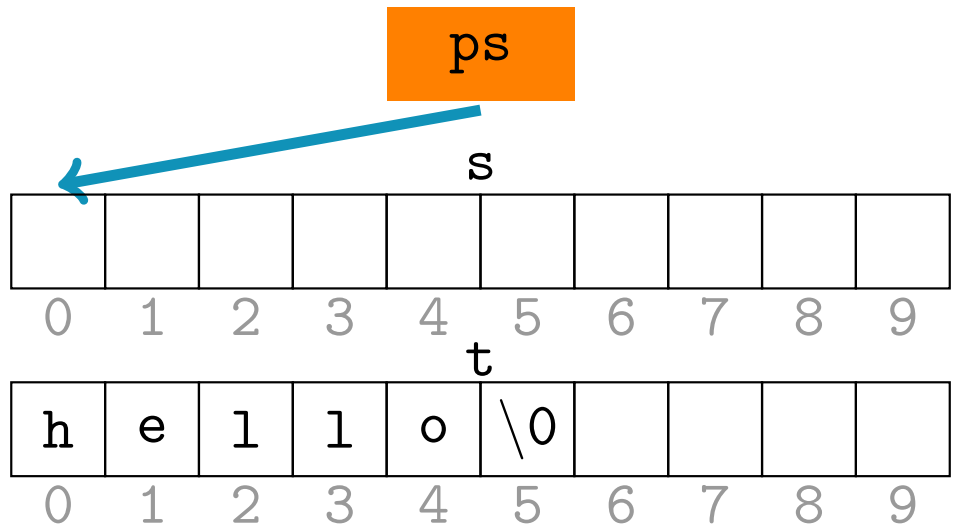
```
char t[10]="hello", s[10];/*copy t to s - pointers*/
char *ps;
```



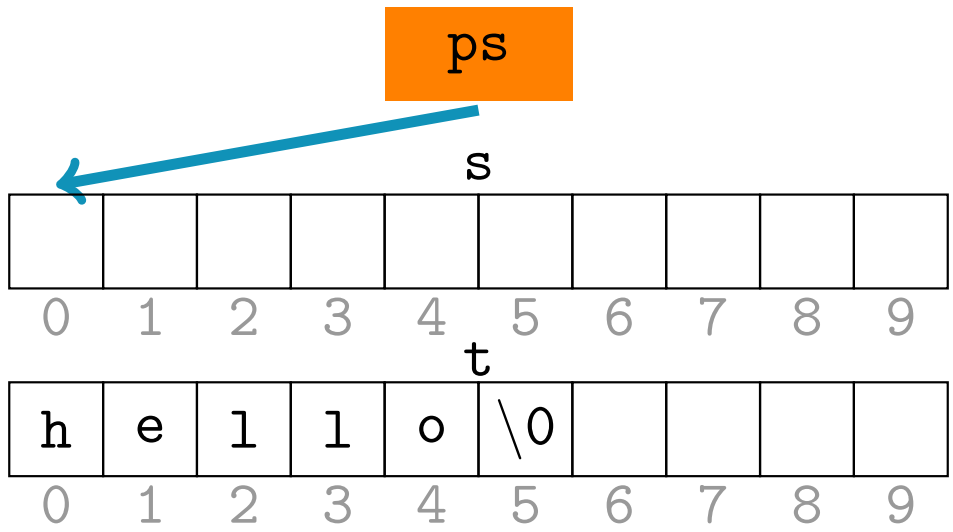
```
char t[10]="hello", s[10];/*copy t to s - pointers*/
char *ps = s;
```



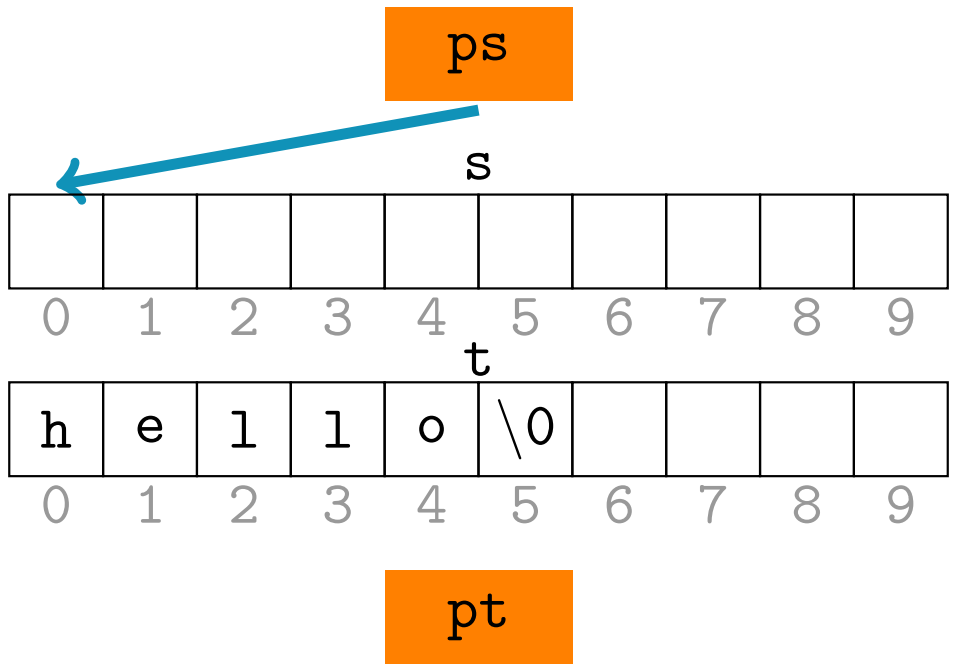
```
char t[10]="hello", s[10];/*copy t to s - pointers*/
char *ps = s;
```



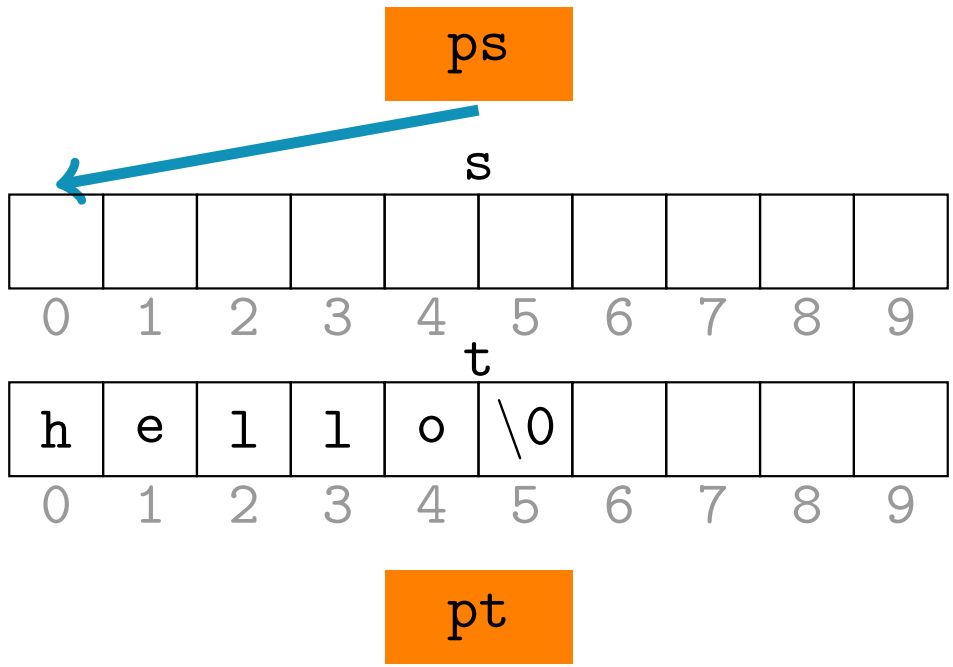
```
char t[10]="hello", s[10];/*copy t to s - pointers*/
char *ps = s, *pt;
```



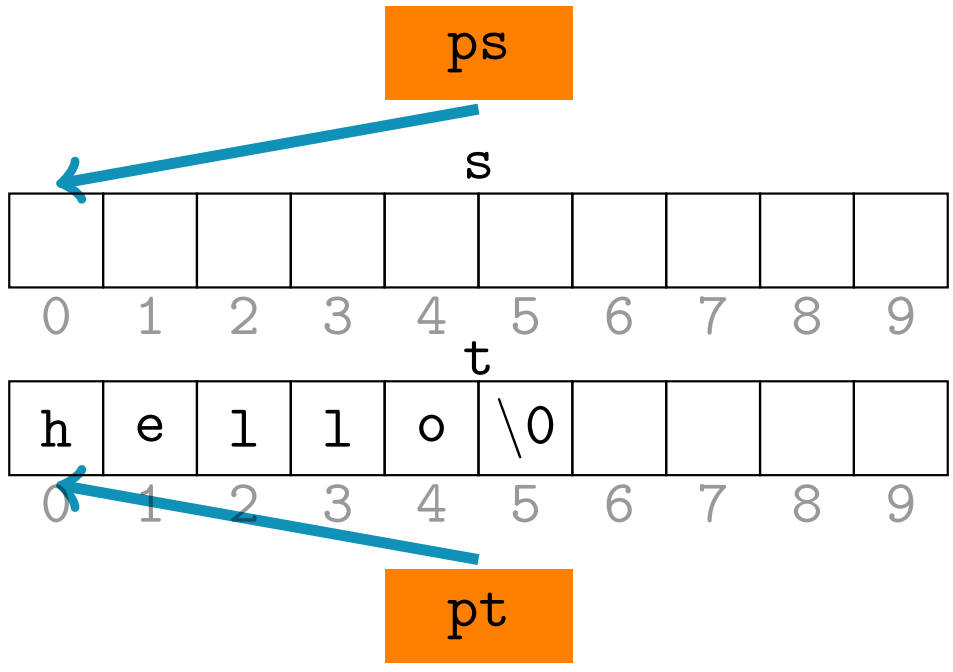
```
char t[10]="hello", s[10];/*copy t to s - pointers*/
char *ps = s, *pt;
```



```
char t[10]="hello", s[10];/*copy t to s - pointers*/
char *ps = s, *pt = t;
```



```
char t[10]="hello", s[10];/*copy t to s - pointers*/
char *ps = s, *pt = t;
```



```
char t[10]="hello", s[10];/*copy t to s - pointers*/
char *ps = s, *pt = t;

while (*ps++ = *pt++);
```