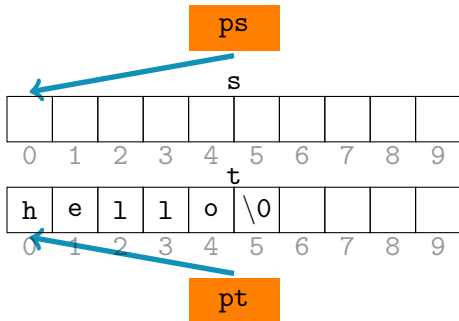
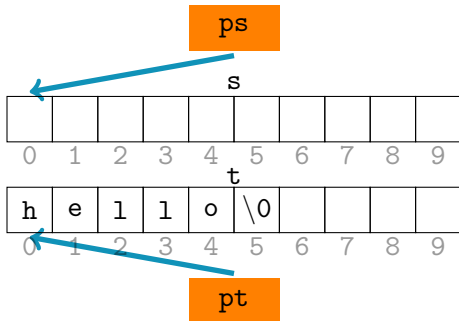


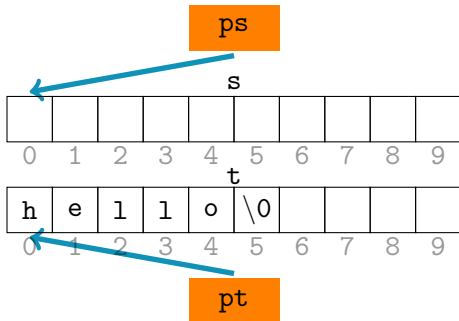
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt */  
while (*ps++ = *pt++);
```



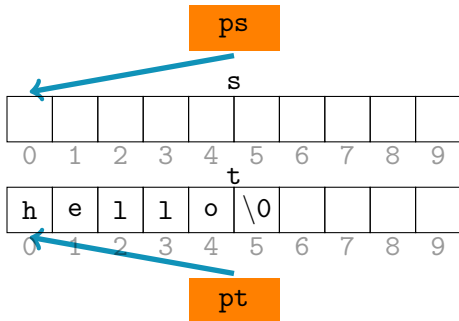
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++ */  
while (*ps++ = *pt++);
```



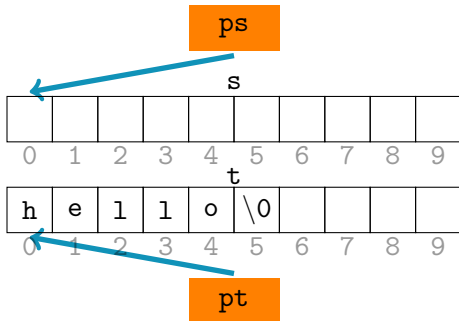
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++ */  
while (*ps++ = *pt++);
```



```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition? */  
while (*ps++ = *pt++);
```

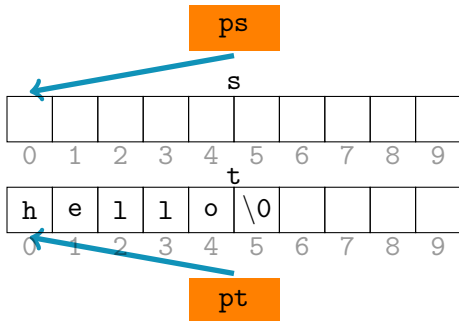


```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++);
```



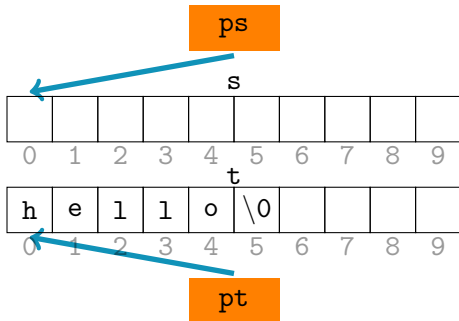
```
int a, b;
```

```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++);
```



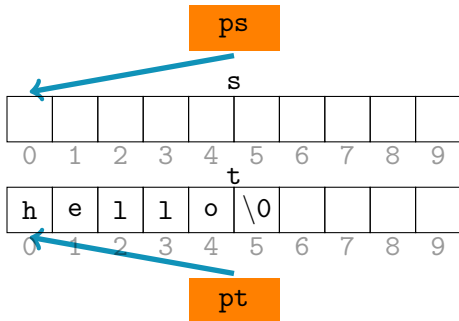
```
int a, b;  
b = 10;
```

```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++);
```



```
int a, b;  
b = 10;  
printf("%d", a = b);
```

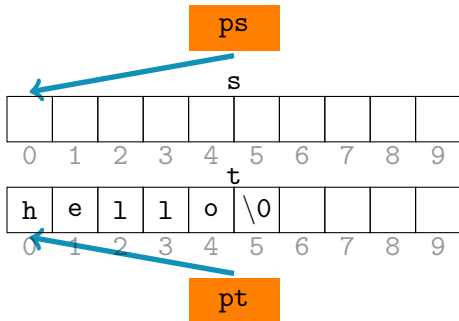
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++);
```



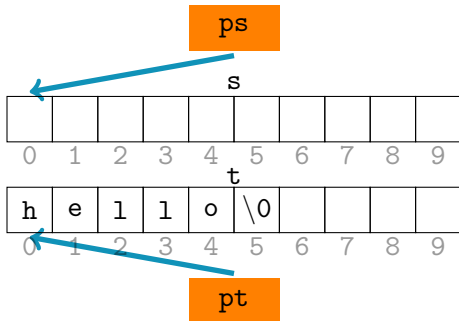
```
int a, b;  
b = 10;  
printf("%d", a = b); → 10
```



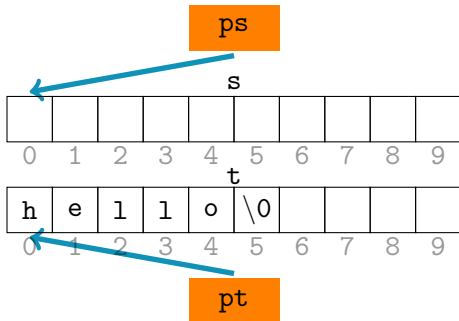
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++);
```



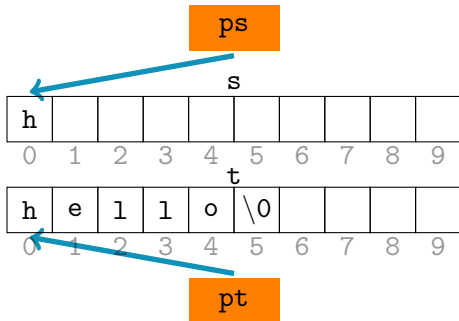
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++);
```



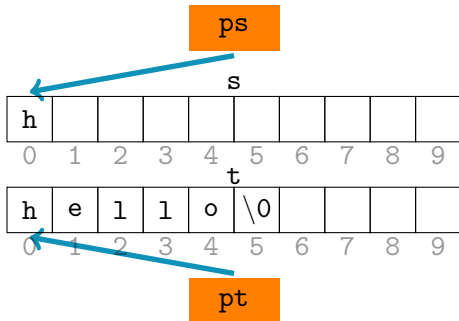
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++);
```



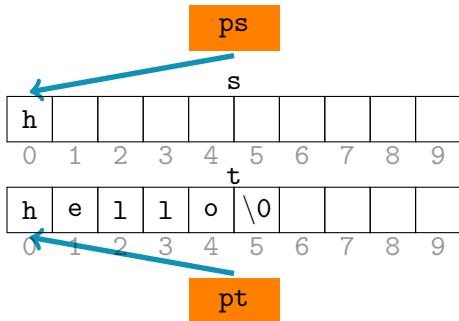
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++);
```



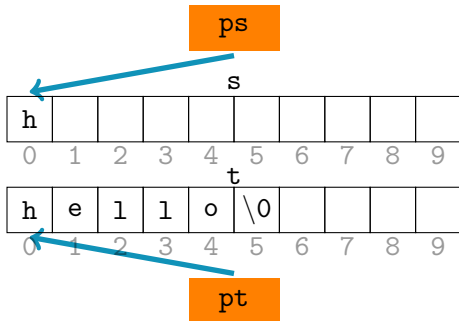
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(*ps = *pt) */
```



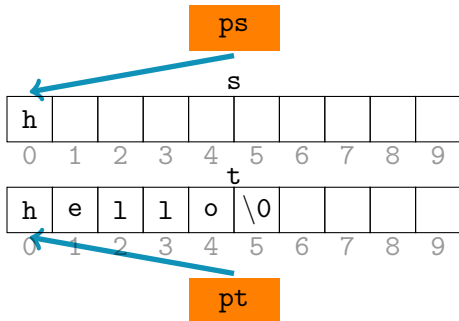
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while('h') */
```



```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(true) */
```

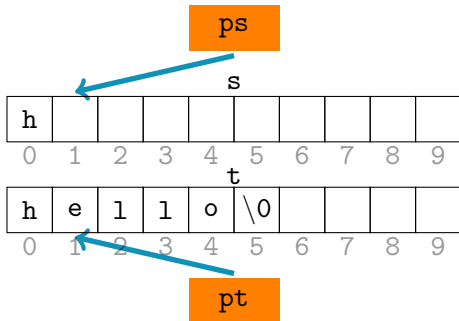


```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(true) */
```

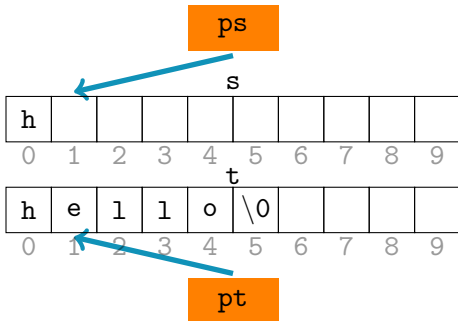




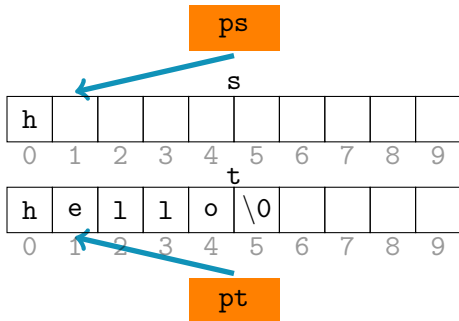
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(true) */
```



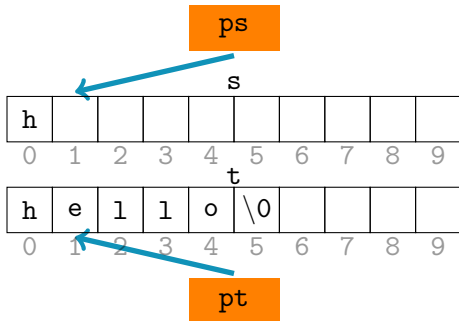
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(true) */
```



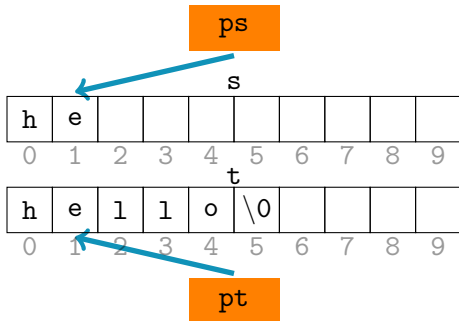
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++);
```



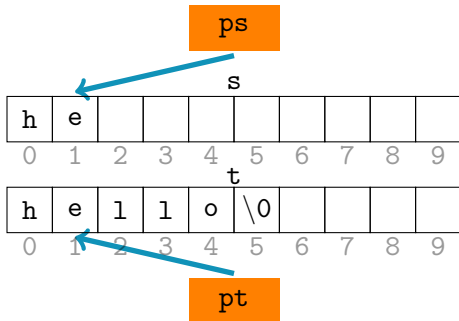
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++);
```



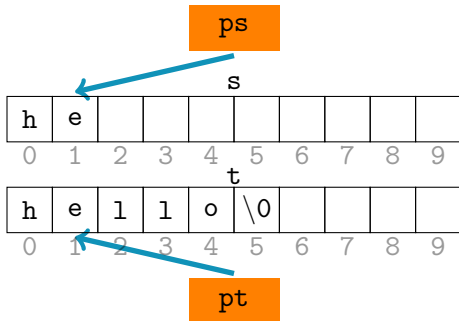
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++);
```



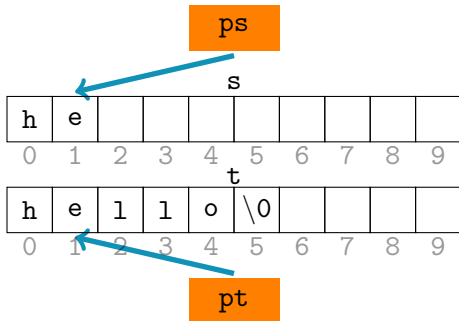
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(*ps = *pt) */
```



```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while('e') */
```

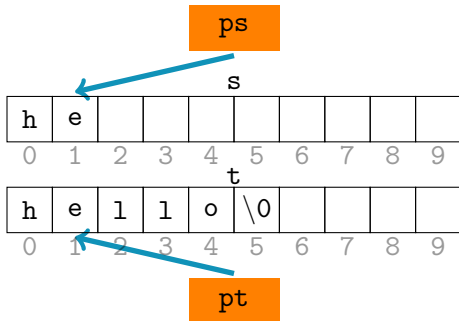


```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(true) */
```

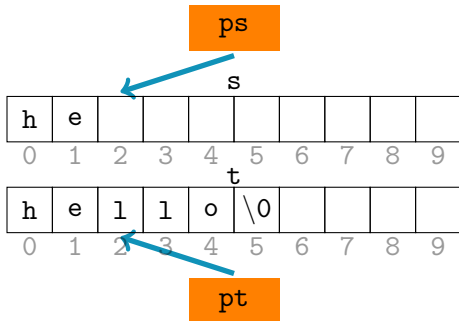




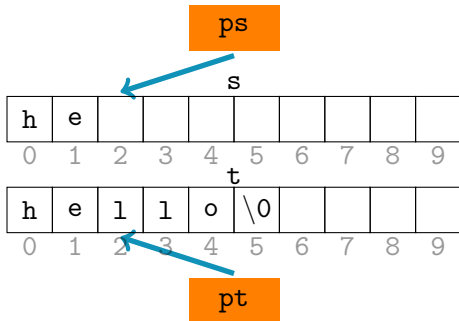
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(true) */
```



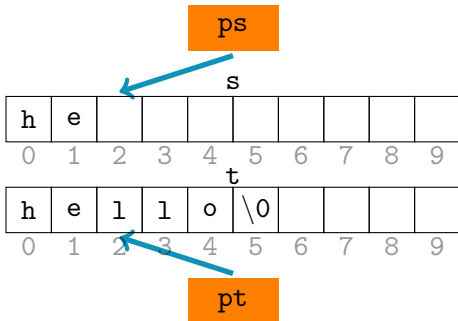
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(true) */
```



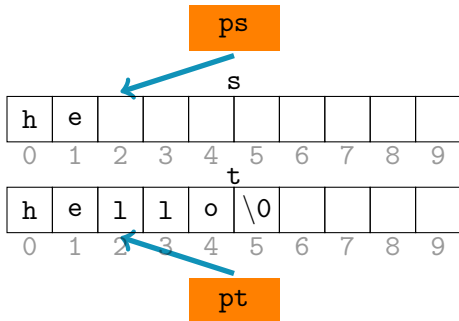
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(true) */
```



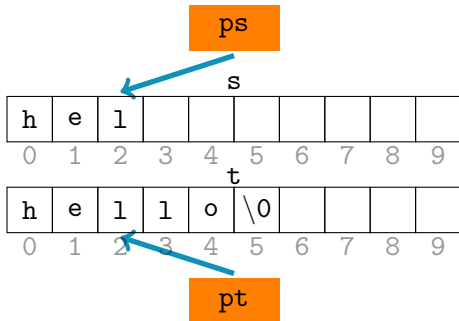
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++);
```



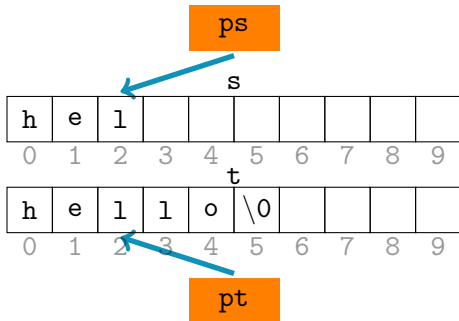
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++);
```



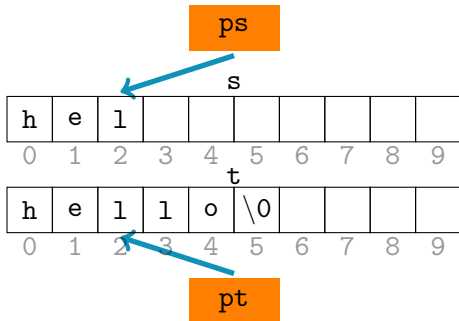
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++);
```



```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(*ps = *pt) */
```

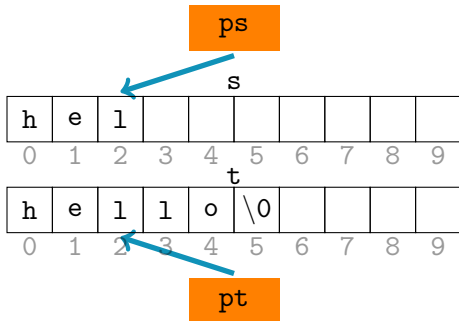


```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while('1') */
```

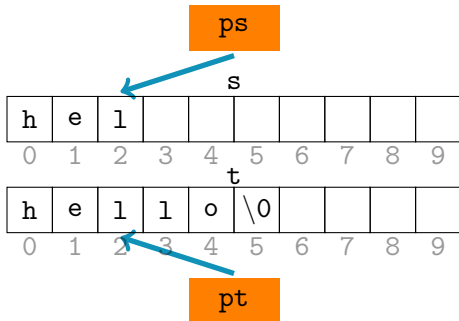




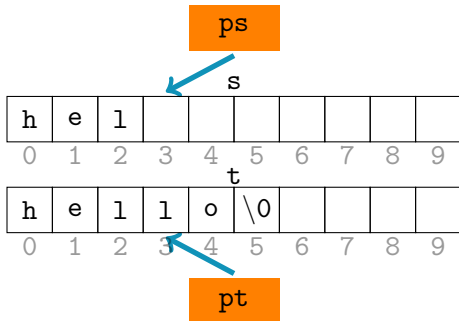
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(true) */
```



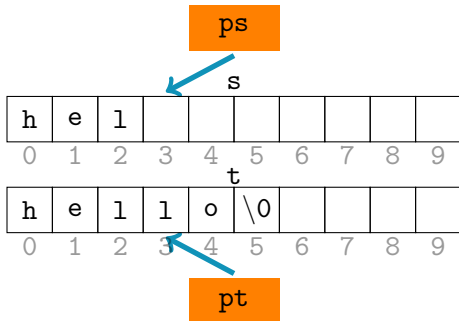
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(true) */
```



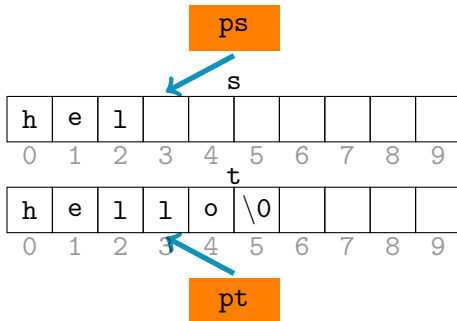
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(true) */
```



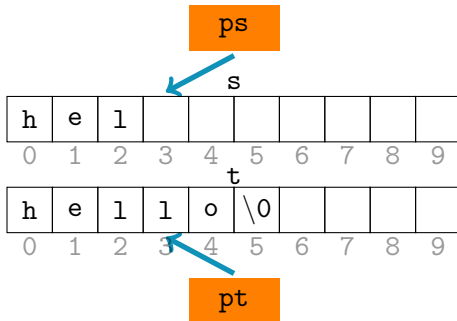
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(true) */
```



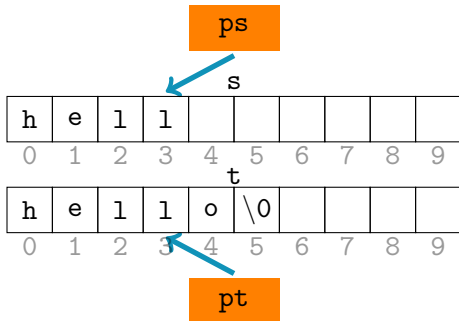
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++);
```



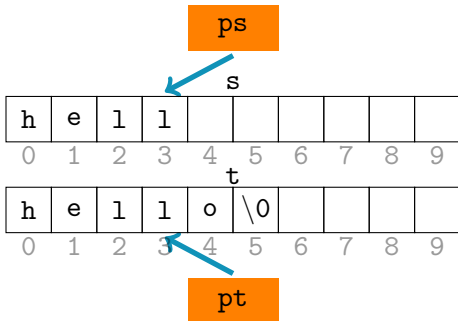
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++);
```



```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++);
```

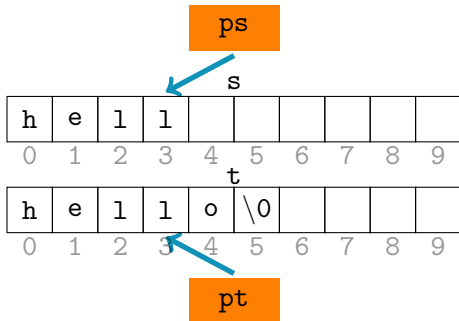


```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(*ps = *pt) */
```

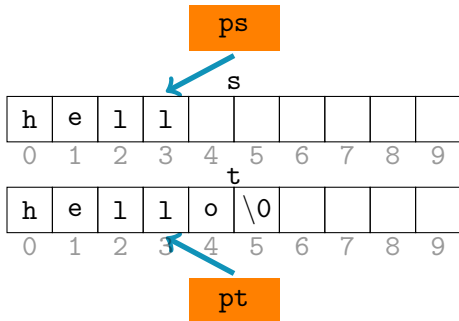




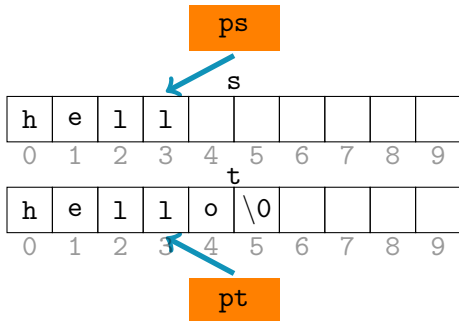
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while('1') */
```



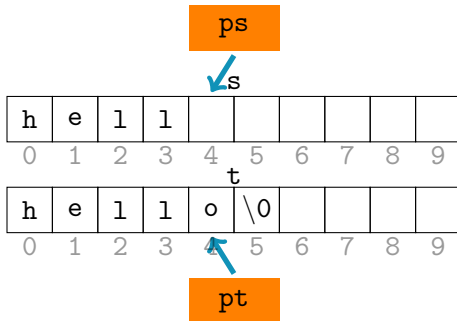
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(true) */
```



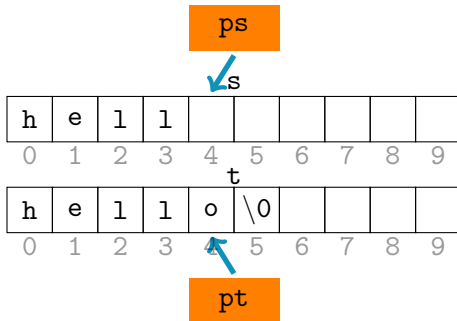
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(true) */
```



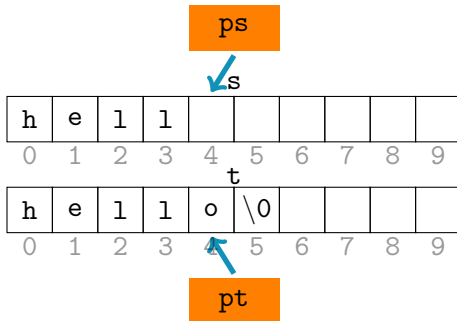
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(true) */
```



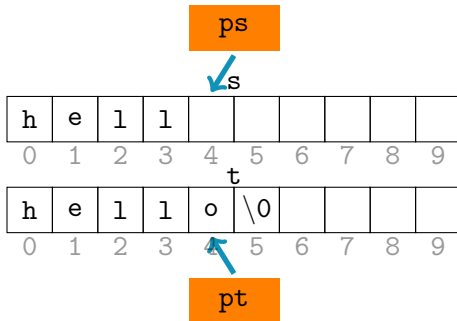
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(true) */
```



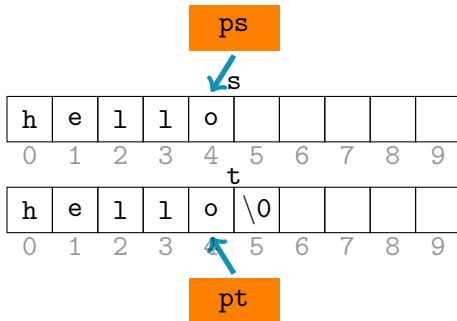
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++);
```



```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++);
```

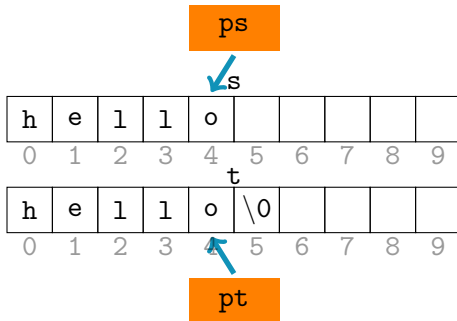


```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++);
```

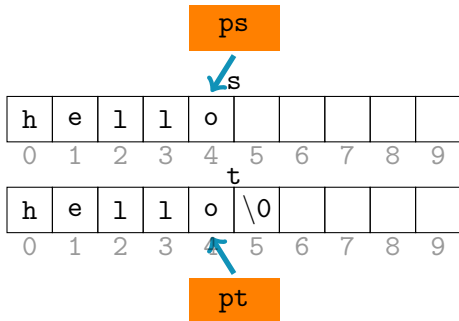




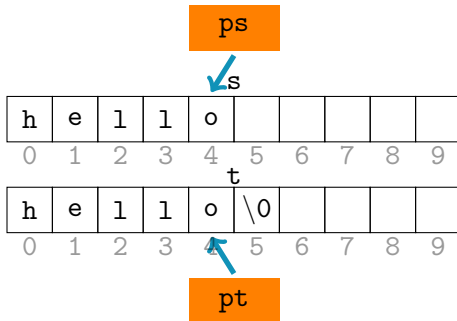
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(*ps = *pt) */
```



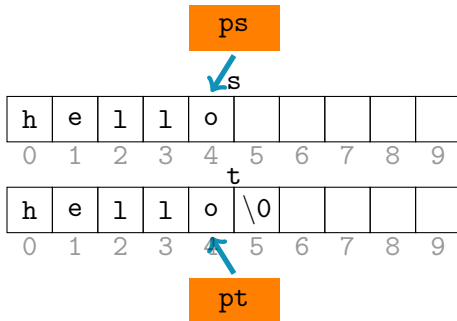
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while('o') */
```



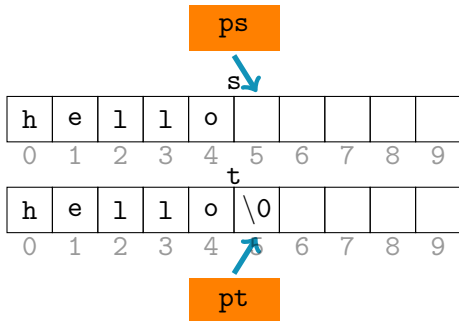
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(true) */
```



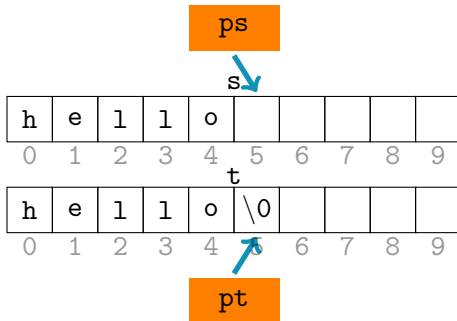
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(true) */
```



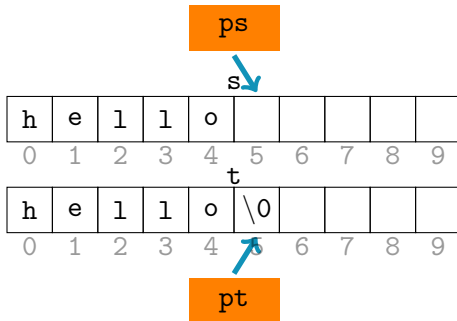
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(true) */
```



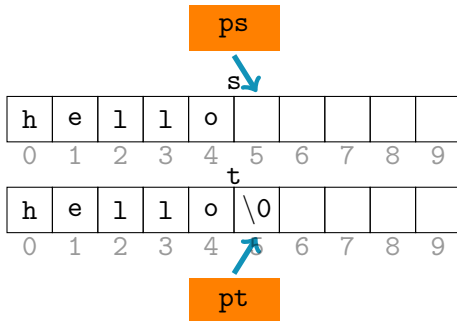
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(true) */
```



```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++);
```

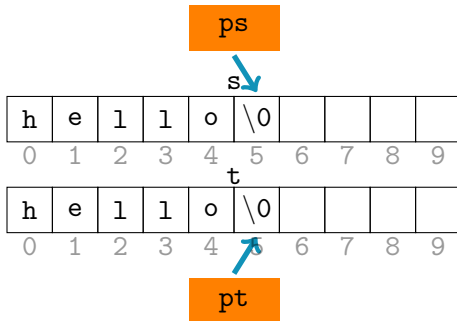


```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++);
```

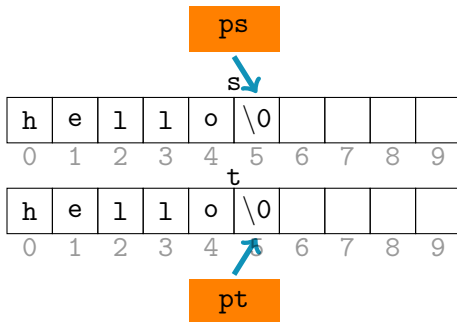




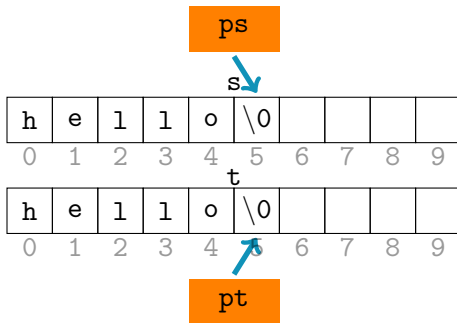
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++);
```



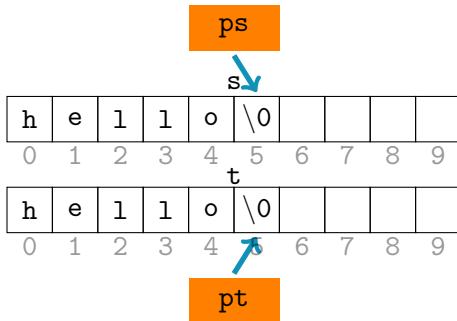
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(*ps = *pt) */
```



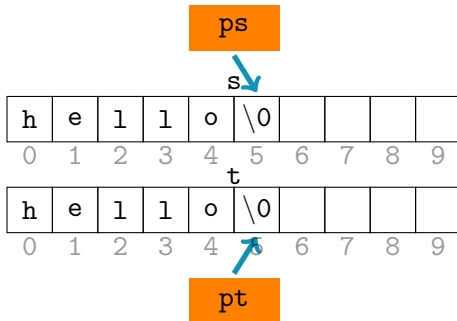
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while('\'0') */
```



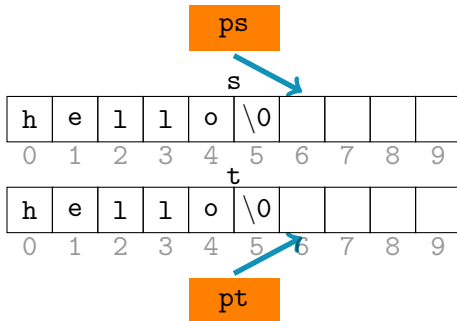
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(false) */
```



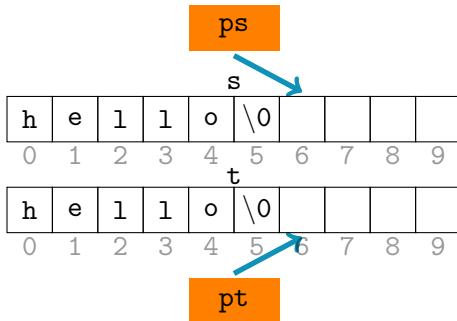
```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(false) */
```



```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(false) */
```



```
char t[10]="hello", s[10];/*copy t to s - pointers*/  
char *ps = s, *pt = t;  
/* *ps = *pt, ps++, pt++; condition - value of *ps */  
while (*ps++ = *pt++); /* while(false) */
```



Done!

## String Copy

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



## String Copy

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

### Comments

## String Copy

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

### Comments

If addresses of s and t are known, we can copy the string.

## String Copy

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

### Comments

If addresses of s and t are known, we can copy the string.

## String Copy

```
char t[] = "hello", s[10]; /*copy t to s -pointers*/  
char *ps = s, *pt = t;  
void strcpy(char *ps, char *pt) {  
    while(*ps++ = *pt++);  
}
```

### Comments

If addresses of s and t are known, we can copy the string.

## String Copy

```
void strcpy(char *ps, char *pt) {  
    while(*ps++ = *pt++);  
}
```

## String Copy

```
void strcpy(char *ps, char *pt) {  
    while(*ps++ = *pt++);  
}  
char t[] = "Hello, world!";
```

## String Copy

```
void strcpy(char *ps, char *pt) {  
    while(*ps++ = *pt++);  
}  
char t[] = "Hello, world!";  
char s[100];
```

## String Copy

```
void strcpy(char *ps, char *pt) {  
    while(*ps++ = *pt++);  
}  
char t[] = "Hello, world!";  
char s[100];  
strcpy(s,t);
```



## String Copy

```
void strcpy(char *ps, char *pt) {  
    while(*ps++ = *pt++);  
}  
char t[] = "Hello, world!";  
char s[100];  
strcpy(s,t);  
printf("%s", s);
```

## String Copy

```
void strcpy(char *ps, char *pt) {  
    while(*ps++ = *pt++);  
}  
char t[] = "Hello, world!";  
char s[100];  
strcpy(s,t);  
printf("%s", s); → Hello, world!
```

# Structures

# Structures

## Definition

Collection of one or more variables, grouped together for convenient handling.

# Structures

## Definition

Collection of one or more variables, grouped together for convenient handling.

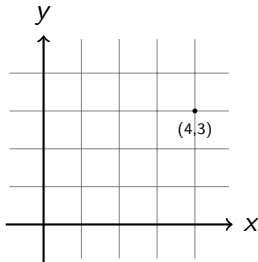
## Why?

# Structures

## Definition

Collection of one or more variables, grouped together for convenient handling.

## Why?

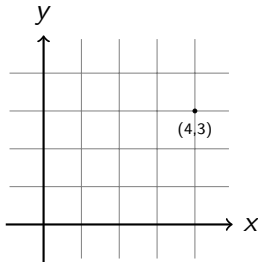


# Structures

## Definition

Collection of one or more variables, grouped together for convenient handling.

## Why?



## Task

A variable to capture a point on the plane.

```
struct point {  
    int x;  
    int y;  
};
```



```
struct point {  
    int x;  
    int y;  
};
```

Semantics

```
struct point {  
    int x;  
    int y;  
};
```

## Semantics

I have declared a structure

```
struct point {  
    int x;  
    int y;  
};
```

## Semantics

I have declared a structure

```
struct point {  
    int x;  
    int y;  
};
```

## Semantics

I have declared a structure by the name point

```
struct point {  
    int x;  
    int y;  
};
```

## Semantics

I have declared a structure by the name point and it will store two integers.

```
struct point {  
    int x; /* x coordinate */  
    int y; /* y coordinate */  
};
```

## Semantics

I have declared a structure by the name point and it will store two integers.

```
struct point {  
    int x; /* x coordinate */  
    int y; /* y coordinate */  
};
```

## Semantics

I have declared a structure by the name point and it will store two integers.

## Fact

```
struct point {  
    int x; /* x coordinate */  
    int y; /* y coordinate */  
};
```

## Semantics

I have declared a structure by the name point and it will store two integers.

## Fact

- “struct point” is a new data type.



```
struct point {  
    int x; /* x coordinate */  
    int y; /* y coordinate */  
};
```

## Semantics

I have declared a structure by the name point and it will store two integers.

## Fact

- “struct point” is a new data type.
- I can declare variables of type “struct point”.

```
struct point {  
    int x; /* x coordinate */  
    int y; /* y coordinate */  
};
```

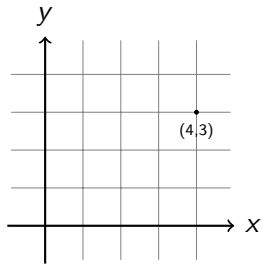
## Semantics

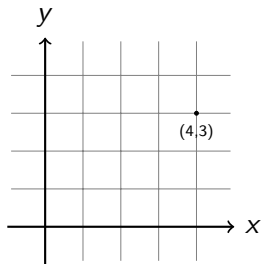
I have declared a structure by the name point and it will store two integers.

## Fact

- “struct point” is a new data type.
- I can declare variables of type “struct point”.

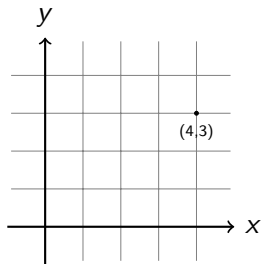
```
struct point pt;
```





## Data type

```
struct point {  
    int x; /* x coordinate */  
    int y; /* y coordinate */  
};
```

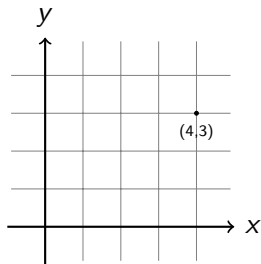


## Data type

```
struct point {  
    int x; /* x coordinate */  
    int y; /* y coordinate */  
};
```

## Variable

```
struct point pt;
```



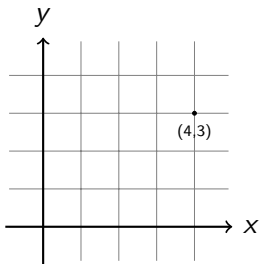
## Data type

```
struct point {  
    int x; /* x coordinate */  
    int y; /* y coordinate */  
};
```

## Initialisation

## Variable

```
struct point pt;
```



## Data type

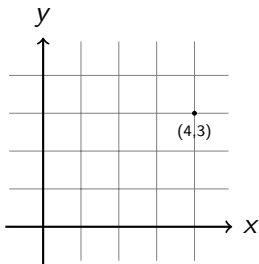
```
struct point {  
    int x; /* x coordinate */  
    int y; /* y coordinate */  
};
```

## Variable

```
struct point pt;
```

## Initialisation

- Set x-coordinate of pt.



## Data type

```
struct point {  
    int x; /* x coordinate */  
    int y; /* y coordinate */  
};
```

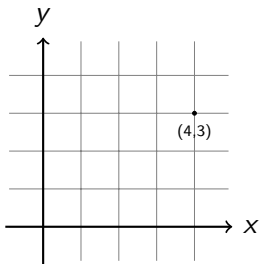
## Variable

```
struct point pt;
```

## Initialisation

- Set x-coordinate of pt.  
pt.x = 4;





## Data type

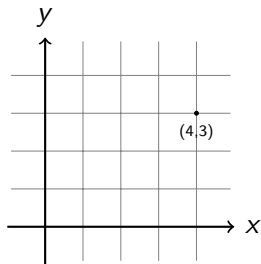
```
struct point {  
    int x; /* x coordinate */  
    int y; /* y coordinate */  
};
```

## Variable

```
struct point pt;
```

## Initialisation

- Set x-coordinate of pt.  
pt.x = 4;
- Set y-coordinate of pt.



## Data type

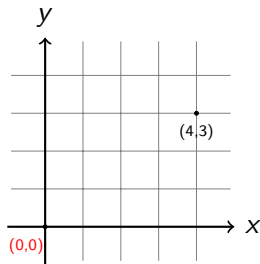
```
struct point {  
    int x; /* x coordinate */  
    int y; /* y coordinate */  
};
```

## Variable

```
struct point pt;
```

## Initialisation

- Set x-coordinate of pt.  
pt.x = 4;
- Set y-coordinate of pt.  
pt.y = 3;



## Data type

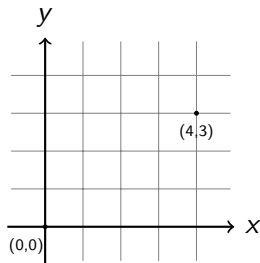
```
struct point {  
    int x; /* x coordinate */  
    int y; /* y coordinate */  
};
```

## Variable

```
struct point pt;
```

## Initialisation

- Set x-coordinate of pt.  
pt.x = 4;
- Set y-coordinate of pt.  
pt.y = 3;



## Data type

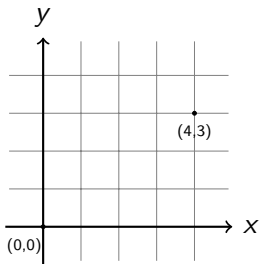
```
struct point {  
    int x; /* x coordinate */  
    int y; /* y coordinate */  
};
```

## Variable

```
struct point pt;
```

## Initialisation

- Set x-coordinate of pt.  
pt.x = 4;
- Set y-coordinate of pt.  
pt.y = 3;
- **struct point** or;



## Data type

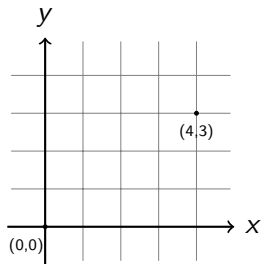
```
struct point {  
    int x; /* x coordinate */  
    int y; /* y coordinate */  
};
```

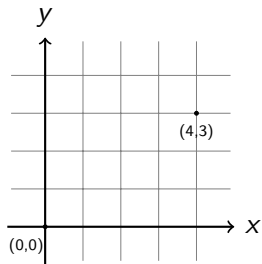
## Variable

```
struct point pt;
```

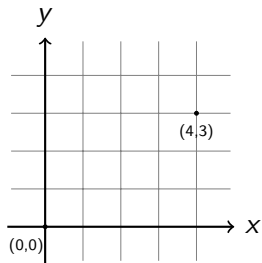
## Initialisation

- Set x-coordinate of pt.  
pt.x = 4;
- Set y-coordinate of pt.  
pt.y = 3;
- struct point or = {0,0};





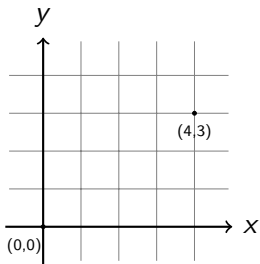
Distance



Distance

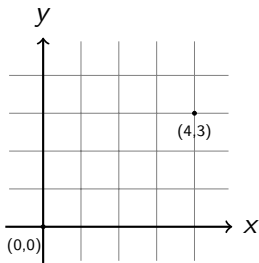
$(\text{pt.x} - \text{or.x})$





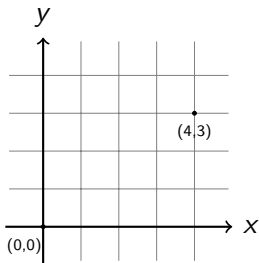
Distance

$$(pt.x - or.x) * (pt.x - or.x)$$



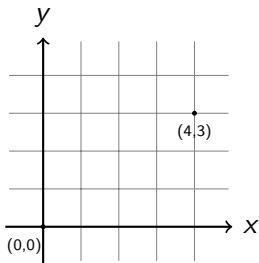
## Distance

$$\sqrt{(pt.x - or.x)^2 + (pt.y - or.y)^2}$$



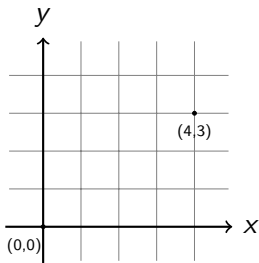
## Distance

```
sqrt((pt.x - or.x) * (pt.x - or.x) + (pt.y  
- or.y) * (pt.y - or.y));
```



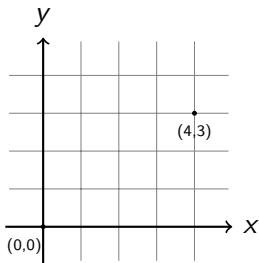
## Distance

```
float d = sqrt((pt.x - or.x) * (pt.x - or.x) + (pt.y  
- or.y) * (pt.y - or.y));
```



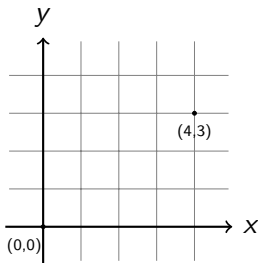
## Distance

```
float d = sqrt((pt.x - or.x) * (pt.x - or.x) + (pt.y  
- or.y) * (pt.y - or.y)); /* include math.h */
```



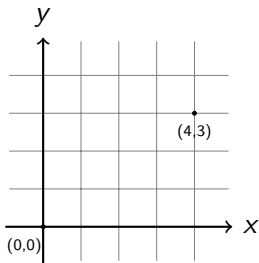
## Distance

```
float d = sqrt((pt.x - or.x) * (pt.x - or.x) + (pt.y  
- or.y) * (pt.y - or.y)); /* include math.h */  
printf("First point :  %d %d", pt.x, pt.y);
```



## Distance

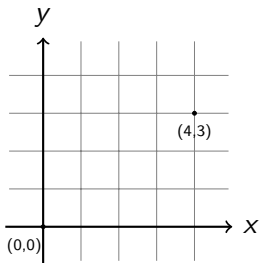
```
float d = sqrt((pt.x - or.x) * (pt.x - or.x) + (pt.y  
- or.y) * (pt.y - or.y)); /* include math.h */  
printf("First point :  %d %d", pt.x, pt.y); → 4 3
```



## Distance

```
float d = sqrt((pt.x - or.x) * (pt.x - or.x) + (pt.y  
- or.y) * (pt.y - or.y)); /* include math.h */  
printf("First point :  %d %d", pt.x, pt.y); → 4 3  
printf("Second point :  %d %d", or.x, or.y);
```



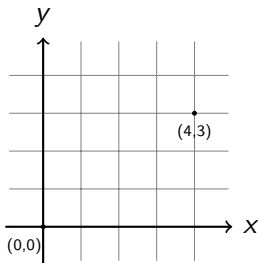


## Distance

```
float d = sqrt((pt.x - or.x) * (pt.x - or.x) + (pt.y  
- or.y) * (pt.y - or.y)); /* include math.h */
```

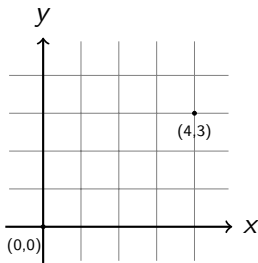
```
printf("First point :  %d %d", pt.x, pt.y); → 4 3
```

```
printf("Second point :  %d %d", or.x, or.y); → 0 0
```



## Distance

```
float d = sqrt((pt.x - or.x) * (pt.x - or.x) + (pt.y  
- or.y) * (pt.y - or.y)); /* include math.h */  
printf("First point :  %d %d", pt.x, pt.y); → 4 3  
printf("Second point :  %d %d", or.x, or.y); → 0 0  
printf("Distance :  %f", d);
```



## Distance

```
float d = sqrt((pt.x - or.x) * (pt.x - or.x) + (pt.y  
- or.y) * (pt.y - or.y)); /* include math.h */  
printf("First point :  %d %d", pt.x, pt.y); → 4 3  
printf("Second point :  %d %d", or.x, or.y); → 0 0  
printf("Distance :  %f", d); → 5.000000
```

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

```
int main(void)
{
```

```
    return 0;
}
```

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

```
int main(void)
{
    struct point {
        int x;
        int y;
    };

```

```
    return 0;
}
```

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

```
int main(void)
```

```
{
```

```
    struct point {
```

```
        int x;
```

```
        int y;
```

```
    };
```

```
    struct point pt = {4,3};
```

```
    return 0;
```

```
}
```

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

```
int main(void)
```

```
{
```

```
    struct point {
```

```
        int x;
```

```
        int y;
```

```
    };
```

```
    struct point pt = {4,3};
```

```
    struct point or = {0,0};
```

```
    return 0;
```

```
}
```

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

```
int main(void)
```

```
{
```

```
    struct point {
```

```
        int x;
```

```
        int y;
```

```
    };
```

```
    struct point pt = {4,3};
```

```
    struct point or = {0,0};
```

```
    float d = sqrt((pt.x - or.x) * (pt.x - or.x) +  
(pt.y - or.y) * (pt.y - or.y));
```

```
    return 0;
```

```
}
```



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

int main(void)
{
    struct point {
        int x;
        int y;
    };

    struct point pt = {4,3};
    struct point or = {0,0};

    float d = sqrt((pt.x - or.x) * (pt.x - or.x) +
(pt.y - or.y) * (pt.y - or.y));

    return 0;
}
```

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

int main(void)
{
    struct point {
        int x;
        int y;
    };

    struct point pt = {4,3};
    struct point or = {0,0};

    float d = sqrt((pt.x - or.x) * (pt.x - or.x) +
(pt.y - or.y) * (pt.y - or.y));

    printf("Distance :  %f", d);

    return 0;
}
```

```
dist()/* distance function */  
{  
}
```

```
dist(struct point pt1, struct point pt2)
{
}
```

```
dist(struct point pt1, struct point pt2)
{
    float d;
}
```

```
dist(struct point pt1, struct point pt2)
{
    float d;
    d = sqrt((pt1.x - pt2.x) * (pt1.x - pt2.x) + (pt1.y
- pt2.y) * (pt1.y - pt2.y));
}
```

```
dist(struct point pt1, struct point pt2)
{
    float d;
    d = sqrt((pt1.x - pt2.x) * (pt1.x - pt2.x) + (pt1.y
- pt2.y) * (pt1.y - pt2.y));
}
```

```
dist(struct point pt1, struct point pt2)
{
    float d;
    d = sqrt((pt1.x - pt2.x) * (pt1.x - pt2.x) + (pt1.y
- pt2.y) * (pt1.y - pt2.y));
    return d;
}
```



```
float dist(struct point pt1, struct point pt2)
{
    float d;
    d = sqrt((pt1.x - pt2.x) * (pt1.x - pt2.x) + (pt1.y
- pt2.y) * (pt1.y - pt2.y));
    return d;
}
```

```
#include <stdio.h>
#include <math.h>
float dist(struct point, struct point);
int main(void) {
    return 0;
}
float dist(struct point pt1, struct point pt2)
{
    float d;
    d = sqrt((pt1.x - pt2.x) * (pt1.x - pt2.x) + (pt1.y
- pt2.y) * (pt1.y - pt2.y));
    return d;
}
```

```
#include <stdio.h>
#include <math.h>
float dist(struct point, struct point);
int main(void) {
    struct point {
        int x;
        int y;
    };
    return 0;
}
float dist(struct point pt1, struct point pt2)
{
    float d;
    d = sqrt((pt1.x - pt2.x) * (pt1.x - pt2.x) + (pt1.y
- pt2.y) * (pt1.y - pt2.y));
    return d;
}
```

```
#include <stdio.h>
#include <math.h>
float dist(struct point, struct point);
int main(void) {
    struct point {
        int x;
        int y;
    };
    struct point pt = {4,3}, or = {0,0};
    return 0;
}
float dist(struct point pt1, struct point pt2)
{
    float d;
    d = sqrt((pt1.x - pt2.x) * (pt1.x - pt2.x) + (pt1.y
- pt2.y) * (pt1.y - pt2.y));
    return d;
}
```

```
#include <stdio.h>
#include <math.h>
float dist(struct point, struct point);
int main(void) {
    struct point {
        int x;
        int y;
    };
    struct point pt = {4,3}, or = {0,0};
    dist(pt, or);
    return 0;
}
float dist(struct point pt1, struct point pt2)
{
    float d;
    d = sqrt((pt1.x - pt2.x) * (pt1.x - pt2.x) + (pt1.y
- pt2.y) * (pt1.y - pt2.y));
    return d;
}
```

```
#include <stdio.h>
#include <math.h>
float dist(struct point, struct point);
int main(void) {
    struct point {
        int x;
        int y;
    };
    struct point pt = {4,3}, or = {0,0};
    printf("%f", dist(pt, or));
    return 0;
}
float dist(struct point pt1, struct point pt2)
{
    float d;
    d = sqrt((pt1.x - pt2.x) * (pt1.x - pt2.x) + (pt1.y
- pt2.y) * (pt1.y - pt2.y));
    return d;
}
```

```
#include <stdio.h>
#include <math.h>
float dist(struct point, struct point)
int main(void) {
    struct point {
        int x;
        int y;
    };
    struct point pt = {4,3}, pr = {0,0};
    printf("%f", dist(pt, pr));
    return 0;
}
float dist(struct point pt1, struct point pt2)
{
    float d;
    d = sqrt((pt1.x - pt2.x) * (pt1.x - pt2.x) + (pt1.y
- pt2.y) * (pt1.y - pt2.y));
    return d;
}
```

```
#include <stdio.h>
#include <math.h>
float dist(struct point, struct point);
int main(void) {
    struct point { /* local to main() */
        int x;
        int y;
    };
    struct point pt = {4,3}, or = {0,0};
    printf("%f", dist(pt, or));
    return 0;
}
float dist(struct point pt1, struct point pt2)
{
    float d;
    d = sqrt((pt1.x - pt2.x) * (pt1.x - pt2.x) + (pt1.y
- pt2.y) * (pt1.y - pt2.y));
    return d;
}
```



```
#include <stdio.h>
#include <math.h>
struct point {
    int x;
    int y;
};
float dist(struct point, struct point);
int main(void) {
    struct point pt = {4,3}, or = {0,0};
    printf("%f", dist(pt, or));
    return 0;
}
float dist(struct point pt1, struct point pt2)
{
    float d;
    d = sqrt((pt1.x - pt2.x) * (pt1.x - pt2.x) + (pt1.y
- pt2.y) * (pt1.y - pt2.y));
    return d;
}
```

```
#include <stdio.h>
#include <math.h>
struct point { /* both main() and dist() knows */
    int x;
    int y;
};
float dist(struct point, struct point);
int main(void) {
    struct point pt = {4,3}, or = {0,0};
    printf("%f", dist(pt, or));
    return 0;
}
float dist(struct point pt1, struct point pt2)
{
    float d;
    d = sqrt((pt1.x - pt2.x) * (pt1.x - pt2.x) + (pt1.y
- pt2.y) * (pt1.y - pt2.y));
    return d;
}
```

```
#include <stdio.h>
#include <math.h>
struct point {
    int x;
    int y;
};

int main(void)
{
    struct point pt = {4,3}, or = {0,0};
    printf("%f", dist(pt, or));
    return 0;
}
```

```
#include <stdio.h>
#include <math.h>
struct point {
    int x;
    int y;
};

int main(void)
{
    struct point pt = {4,3}, or = {0,0};
    printf("%f", dist(pt, or));
    return 0;
}
```

```
#include <stdio.h>
#include <math.h>
struct point {
    int x;
    int y;
};

int main(void)
{
    struct point pt = {4,3}, or = {0,0};
    struct point pts[2];
    printf("%f", dist(pt, or));
    return 0;
}
```

```
#include <stdio.h>
#include <math.h>
struct point {
    int x;
    int y;
};

int main(void)
{
    struct point pt = {4,3}, or = {0,0};
    struct point pts[2]; /* array of structs */
    printf("%f", dist(pt, or));
    return 0;
}
```

```
#include <stdio.h>
#include <math.h>
struct point {
    int x;
    int y;
};

int main(void)
{
    struct point pt = {4,3}, or = {0,0};
    struct point pts[2]; /* array of structs */
    pts[0].x = 4;
    printf("%f", dist(pt, or));
    return 0;
}
```

```
#include <stdio.h>
#include <math.h>
struct point {
    int x;
    int y;
};

int main(void)
{
struct point pt = {4,3}, or = {0,0};
    struct point pts[2]; /* array of structs */
    pts[0].x = 4, pts[0].y = 3;
    printf("%f", dist(pt, or));
    return 0;
}
```



```
#include <stdio.h>
#include <math.h>
struct point {
    int x;
    int y;
};

int main(void)
{
struct point pt = {4,3}, or = {0,0};
    struct point pts[2]; /* array of structs */
    pts[0].x = 4, pts[0].y = 3;
    pts[1].x = 0, pts[1].y = 0;
    printf("%f", dist(pt, or));
    return 0;
}
```

```
#include <stdio.h>
#include <math.h>
struct point {
    int x;
    int y;
};

int main(void)
{
    struct point pt = {4,3}, or = {0,0};
    struct point pts[2]; /* array of structs */
    pts[0].x = 4, pts[0].y = 3;
    pts[1].x = 0, pts[1].y = 0;
    printf("%f", dist(pt, or));
    return 0;
}
```

```
#include <stdio.h>
#include <math.h>
struct point {
    int x;
    int y;
};

int main(void)
{
    struct point pt = {4,3}, or = {0,0};
    struct point pts[2]; /* array of structs */
    pts[0].x = 4, pts[0].y = 3;
    pts[1].x = 0, pts[1].y = 0;
    printf("%f", dist(pts[0], pts[1]));
    return 0;
}
```