```
#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;
}
```

```c
#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;
}
```

```c
#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;
}
```

```
printf("%f", dist(pts[0], pts[1]));
printf("%f", dist(&pts[0], &pts[1]));
```

```
/* send addresses of pts[0] and pts[1] */
printf("%f", dist(&pts[0], &pts[1]));
```

```
/* send addresses of pts[0] and pts[1] */
printf("%f", dist(&pts[0], &pts[1]));

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;
}
```

```
/* send addresses of pts[0] and pts[1] */
printf("%f", dist(&pts[0], &pts[1]));

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;
}
```

```
/* send addresses of pts[0] and pts[1] */
printf("%f", dist(&pts[0], &pts[1]));

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;
}
```

Comments

- pt1 should accept an address.

```
/* send addresses of pts[0] and pts[1] */
printf("%f", dist(&pts[0], &pts[1]));

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;
}
```

Comments

- pt1 should accept an address.
- New data type of pt1?

```
/* send addresses of pts[0] and pts[1] */
printf("%f", dist(&pts[0], &pts[1]));

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;
}
```

Comments

- pt1 should accept an address.
- New data type of pt1? → pointer to struct point.

```
/* send addresses of pts[0] and pts[1] */
printf("%f", dist(&pts[0], &pts[1]));

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;
}
```

Comments

- pt1 should accept an address.
- New data type of pt1? → pointer to struct point.
- struct point *pt1;

```
/* send addresses of pts[0] and pts[1] */
printf("%f", dist(&pts[0], &pts[1]));

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;
}
```

Comments

- `pt1` should accept an address.
- New data type of `pt1`? → pointer to `struct point`.
- `struct point *pt1;`

```
/* send addresses of pts[0] and pts[1] */
printf("%f", dist(&pts[0], &pts[1]));

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;
}
```

Comments

- `pt1` should accept an address.
- New data type of `pt1`? → pointer to `struct point`.
- `struct point *pt1;`
- How to access the elements of a structure using pointers?

```
/* send addresses of pts[0] and pts[1] */
printf("%f", dist(&pts[0], &pts[1]));

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;
}
```

Comments

- `pt1` should accept an address.
- New data type of `pt1`? → pointer to `struct point`.
- `struct point *pt1;`
- How to access the elements of a structure using pointers?
- `pt1`

```
/* send addresses of pts[0] and pts[1] */
printf("%f", dist(&pts[0], &pts[1]));

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;
}
```

Comments

- `pt1` should accept an address.
- New data type of `pt1`? → pointer to `struct point`.
- `struct point *pt1;`
- How to access the elements of a structure using pointers?
- `pt1` → address stored `pt1`.

```
/* send addresses of pts[0] and pts[1] */
printf("%f", dist(&pts[0], &pts[1]));

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;
}
```

Comments

- pt1 should accept an address.
- New data type of pt1? → pointer to `struct point`.
- `struct point *pt1;`
- How to access the elements of a structure using pointers?
- `*pt1`

```
/* send addresses of pts[0] and pts[1] */
printf("%f", dist(&pts[0], &pts[1]));

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;
}
```

Comments

- pt1 should accept an address.
- New data type of pt1? → pointer to struct point.
- struct point *pt1;
- How to access the elements of a structure using pointers?
- *pt1 → the struct at that address.

```
/* send addresses of pts[0] and pts[1] */
printf("%f", dist(&pts[0], &pts[1]));

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;
}
```

Comments

- pt1 should accept an address.
- New data type of pt1? → pointer to struct point.
- struct point *pt1;
- How to access the elements of a structure using pointers?
- (*pt1).x

```
/* send addresses of pts[0] and pts[1] */
printf("%f", dist(&pts[0], &pts[1]));

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;
}
```

Comments

- pt1 should accept an address.
- New data type of pt1? → pointer to `struct point`.
- `struct point *pt1;`
- How to access the elements of a structure using pointers?
- `(*pt1).x` → x-coordinate of the structure whose address is stored in pt1.

```
/* send addresses of pts[0] and pts[1] */
printf("%f", dist(&pts[0], &pts[1]));

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;
}
```

Comments

- `pt1` should accept an address.
- New data type of `pt1`? → pointer to `struct point`.
- `struct point *pt1;`
- How to access the elements of a structure using pointers?
- `(*pt1).x` → x-coordinate of the structure whose address is stored in `pt1`.

Recap

Recap

Definition

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

Recap

Definition

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

Defining struct

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

Recap

Definition

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

Defining struct

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

- Keyword for declaration

Recap

Definition

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

Defining `struct`

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

- Keyword for declaration
- Name of structure

Recap

Definition

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

Defining struct

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

- Keyword for declaration
- Name of structure
- Elements of structure

Recap

Definition

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

Defining `struct`

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

- Keyword for declaration
- Name of structure
- Elements of structure
- We have created a new data type called `struct point`.

Recap

Definition

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

Defining struct

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

Recap

Definition

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

Defining struct

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

Variable

```
struct point pt;
```

Recap

Definition

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

Defining `struct`

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

Variable

```
struct point pt;
```

Initialisation

```
pt.x = 4, pt.y = 3;
```

Recap

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

Recap

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

Array

```
struct point pts[2];
```

Recap

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

Array

```
struct point pts[2];
```

Array Initialisation

```
struct point pts[2];
pts[0].x = 4, pts[0].y = 3;
pts[1].x = 0, pts[1].y = 0;
```

Recap

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

Array

```
struct point pts[2];
```

Array Initialisation

```
struct point pts[2] = {{4,3}, {0,0}};
```

Recap

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

Recap

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

Pointers

Recap

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

Pointers

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

Recap

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

Pointers

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

```
struct point *p;
```

Recap

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

Pointers

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

```
struct point *p;
```

```
p = &pt;
```

Recap

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

Pointers

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

```
struct point *p;
```

```
p = &pt;
```

```
(*p).x = 0;
(*p).y = 0;
```

Recap

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

Pointers

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

```
struct point *p;
```

```
p = &pt;
```

```
(*p).x = 0;
(*p).y = 0;
printf("%d %d", pt.x, pt.y);
```

Recap

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

Pointers

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

```
struct point *p;
```

```
p = &pt;
```

```
(*p).x = 0;
(*p).y = 0;
printf("%d %d", pt.x, pt.y); → 0 0
```

```
dist(struct point pt1, struct point pt2) vs
dist(struct point *pt1, struct point *pt2);
dist(struct point pt1, struct point pt2) vs
dist(struct point *pt1, struct point *pt2);
```

Question

Should we pass structures or pointer to structures?

```
dist(struct point pt1, struct point pt2) vs
dist(struct point *pt1, struct point *pt2);
```

Question

Should we pass structures or pointer to structures?

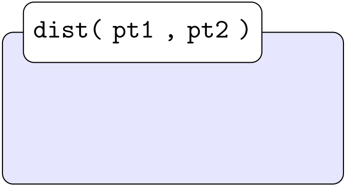
Pass structure by value

```
dist(struct point pt1, struct point pt2) vs
dist(struct point *pt1, struct point *pt2);
```

Question

Should we pass structures or pointer to structures?

Pass structure by value

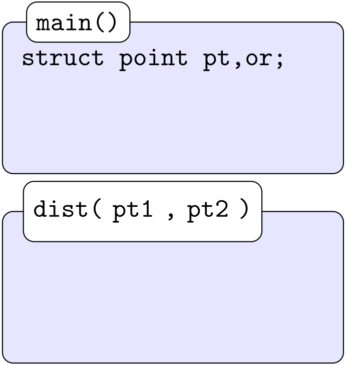


```
dist( pt1 , pt2 )
dist(struct point pt1, struct point pt2) vs
dist(struct point *pt1, struct point *pt2);
```

Question

Should we pass structures or pointer to structures?

Pass structure by value



main()

```
struct point pt,or;
```

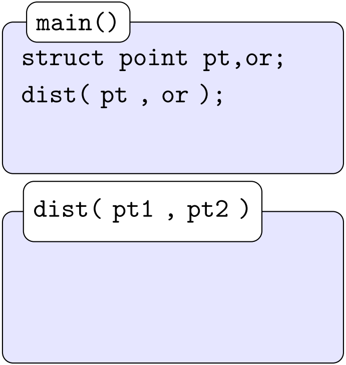
dist(pt1 , pt2)

```
dist(struct point pt1, struct point pt2) vs
dist(struct point *pt1, struct point *pt2);
```

Question

Should we pass structures or pointer to structures?

Pass structure by value



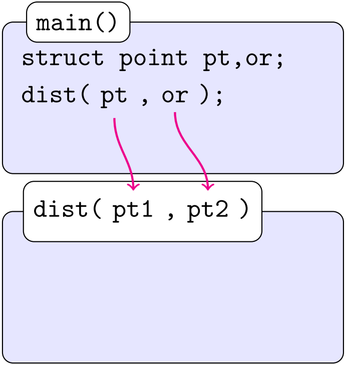
```
main()
struct point pt,or;
dist( pt , or );
```

```
dist( pt1 , pt2 )
dist(struct point pt1, struct point pt2) vs
dist(struct point *pt1, struct point *pt2);
```

Question

Should we pass structures or pointer to structures?

Pass structure by value



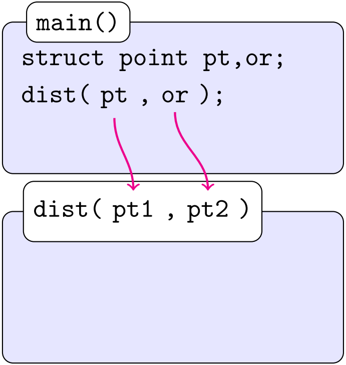
```
main()
struct point pt,or;
dist( pt , or );
```

```
dist( pt1 , pt2 )
dist(struct point pt1, struct point pt2) vs
dist(struct point *pt1, struct point *pt2);
```

Question

Should we pass structures or pointer to structures?

Pass structure by value



```
main()
struct point pt,or;
dist( pt , or );

dist( pt1 , pt2 )
```

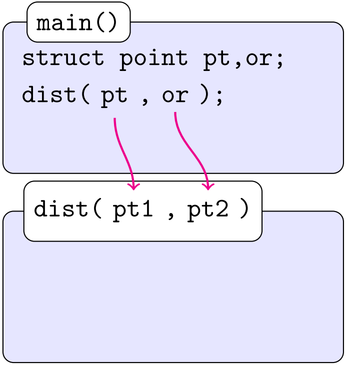
Pass structure by reference

```
dist(struct point pt1, struct point pt2) vs
dist(struct point *pt1, struct point *pt2);
```

Question

Should we pass structures or pointer to structures?

Pass structure by value

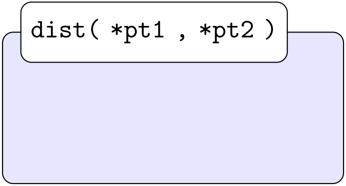


main()

```
struct point pt,or;
dist( pt , or );
```

```
dist( pt1 , pt2 )
```

Pass structure by reference

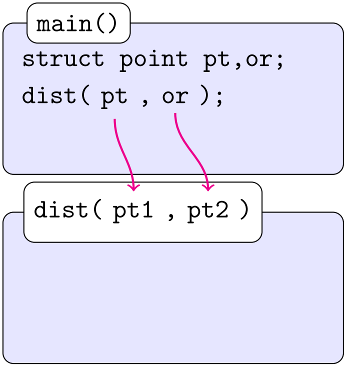


```
dist( *pt1 , *pt2 )
dist(struct point pt1, struct point pt2) vs
dist(struct point *pt1, struct point *pt2);
```

Question

Should we pass structures or pointer to structures?

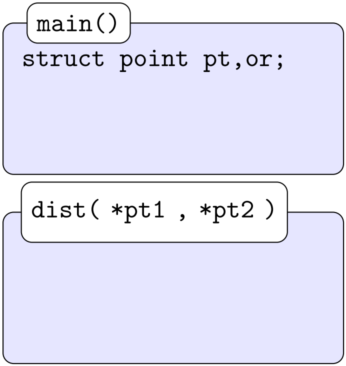
Pass structure by value



```
main()
struct point pt,or;
dist( pt , or );

dist( pt1 , pt2 )
```

Pass structure by reference



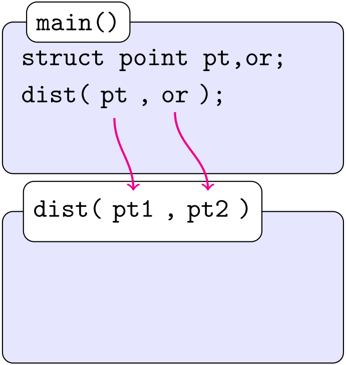
```
main()
struct point pt,or;

dist( *pt1 , *pt2 )
dist(struct point pt1, struct point pt2) vs
dist(struct point *pt1, struct point *pt2);
```

Question

Should we pass structures or pointer to structures?

Pass structure by value

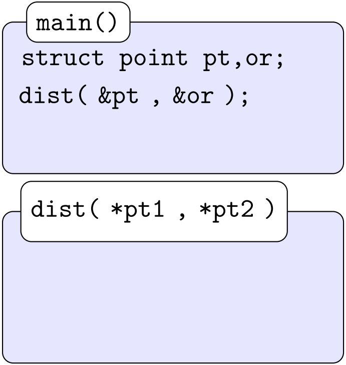


main()

```
struct point pt,or;
dist( pt , or );
```

```
dist( pt1 , pt2 )
```

Pass structure by reference



main()

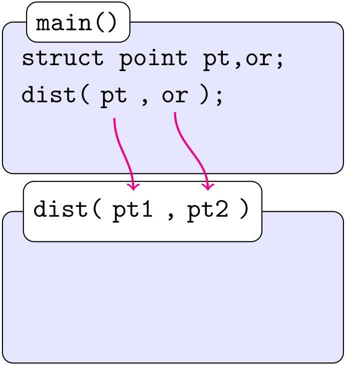
```
struct point pt,or;
dist( &pt , &or );
```

```
dist( *pt1 , *pt2 )
dist(struct point pt1, struct point pt2) vs
dist(struct point *pt1, struct point *pt2);
```

Question

Should we pass structures or pointer to structures?

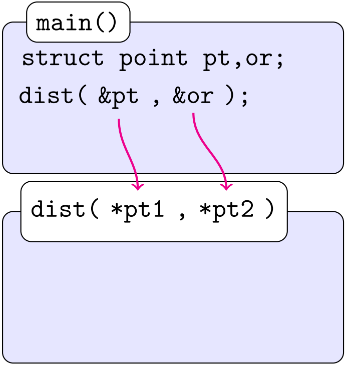
Pass structure by value



```
main()
struct point pt,or;
dist( pt , or );

dist( pt1 , pt2 )
```

Pass structure by reference



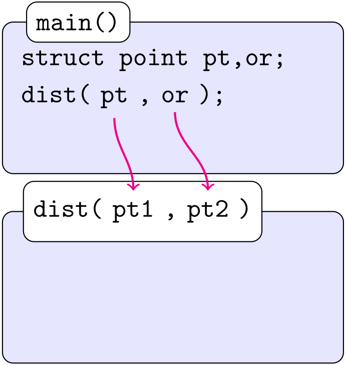
```
main()
struct point pt,or;
dist( &pt , &or );

dist( *pt1 , *pt2 )
dist(struct point pt1, struct point pt2) vs
dist(struct point *pt1, struct point *pt2);
```

Question

Should we pass structures or pointer to structures?

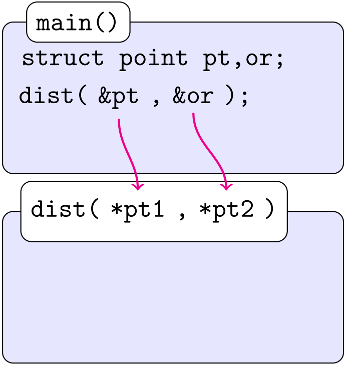
Pass structure by value



```
main()
struct point pt,or;
dist( pt , or );

dist( pt1 , pt2 )
```

Pass structure by reference



```
main()
struct point pt,or;
dist( &pt , &or );

dist( *pt1 , *pt2 )
```

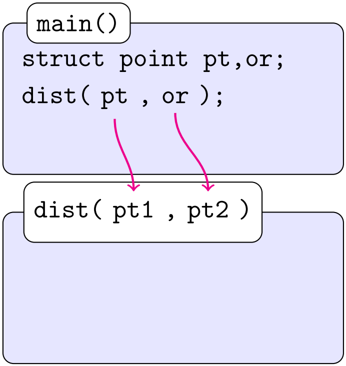
Size of structures might vary.

```
dist(struct point pt1, struct point pt2) vs
dist(struct point *pt1, struct point *pt2);
```

Question

Should we pass structures or pointer to structures?

Pass structure by value

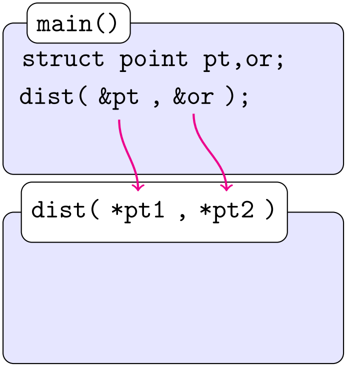


```
main()
struct point pt,or;
dist( pt , or );
```

```
dist( pt1 , pt2 )
```

Size of structures might vary.

Pass structure by reference



```
main()
struct point pt,or;
dist( &pt , &or );
```

```
dist( *pt1 , *pt2 )
```

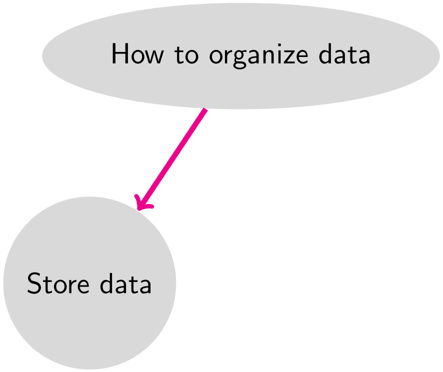
Size of addresses is fixed.

Data Structures

Data Structures

How to organize data

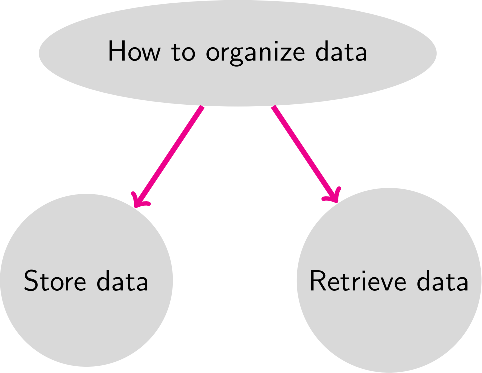
Data Structures



How to organize data

Store data

Data Structures

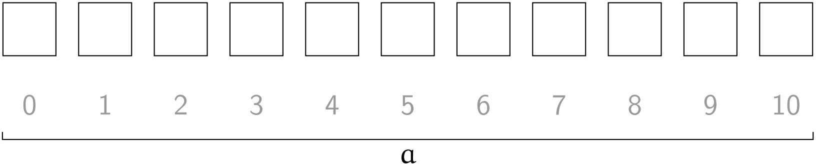


How to organize data

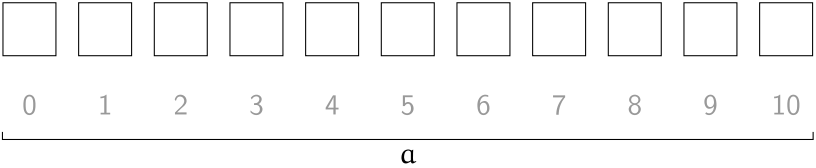
Store data

Retrieve data

Data Structures
- Array

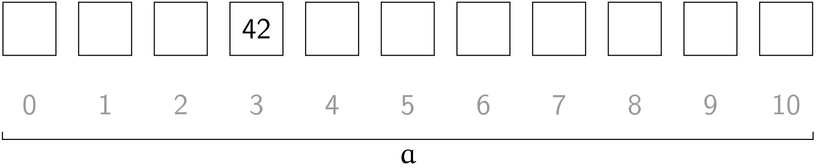


Data Structures
- Array



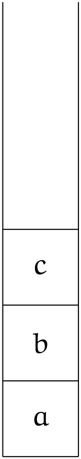
a[3]

Data Structures
- Array



$$a[3] = 42$$

Data Structures - Stacks

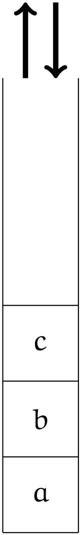


c

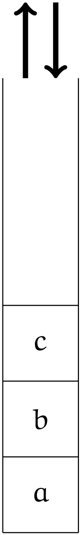
b

a

Data Structures
- Stacks

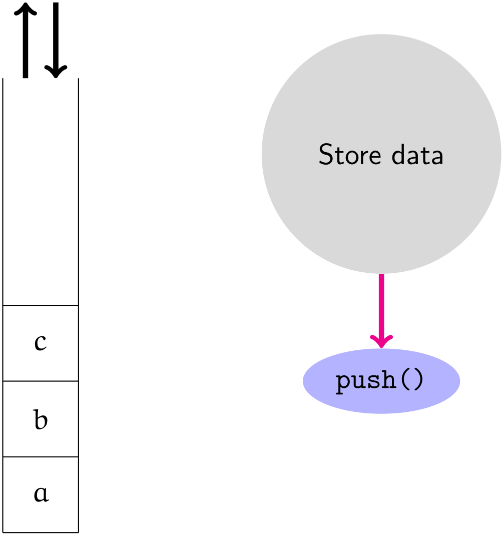


Data Structures
- Stacks

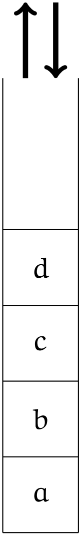


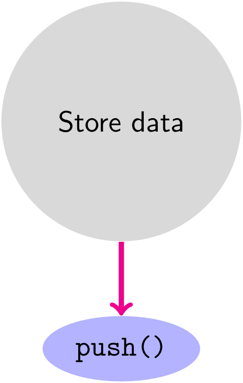
Store data

Data Structures - Stacks



Data Structures - Stacks

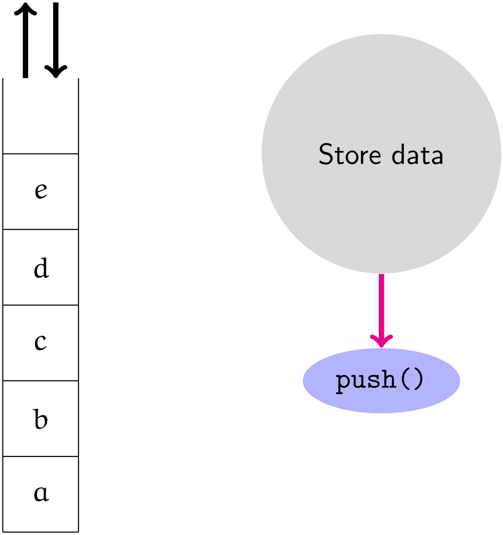




Store data

`push()`

Data Structures - Stacks



e

d

c

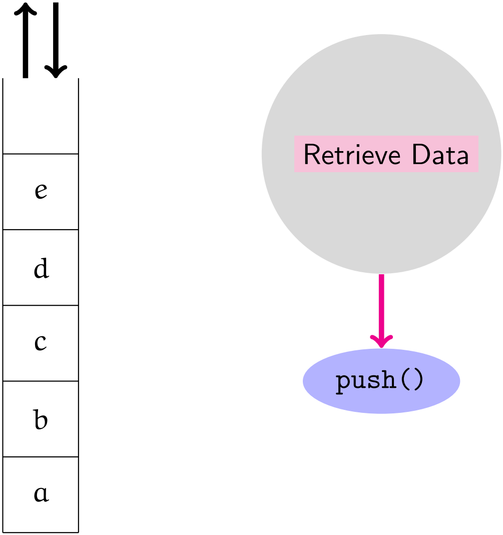
b

a

Store data

push()

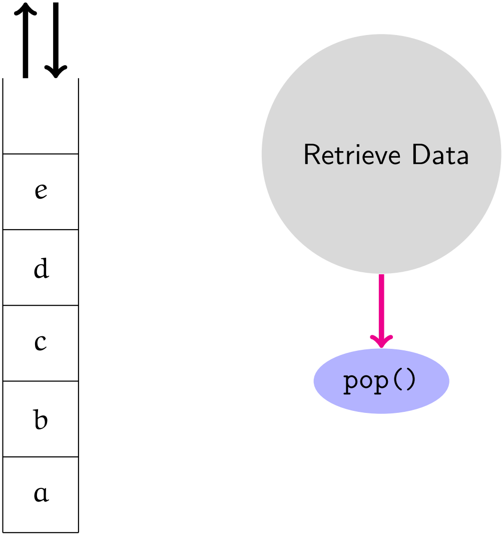
Data Structures - Stacks



Retrieve Data

push()

Data Structures
- Stacks



e

d

c

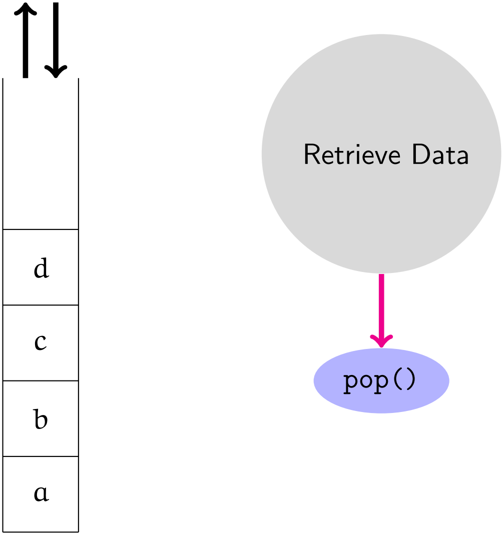
b

a

Retrieve Data

`pop()`

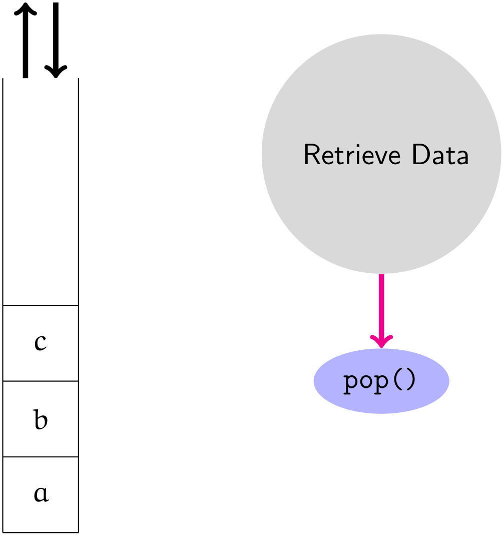
Data Structures - Stacks



Retrieve Data

pop()

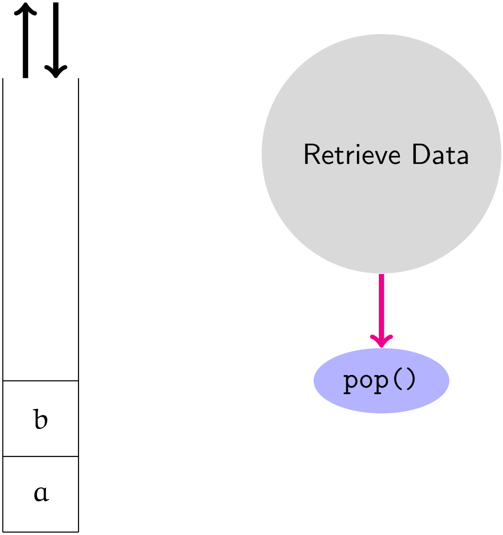
Data Structures
- Stacks



Retrieve Data

`pop()`

Data Structures
- Stacks



Retrieve Data

`pop()`