Pointers in C

Computer Memory Revisited

- Computers store data in memory slots
- Each slot has an *unique address*
- Variables store their values like this:

Addr	**Content**	**Addr**	**Content**	**Addr**	**Content**	**Addr**	**Content**
1000	i: 37	1001	j: 46	1002	k: 58	1003	m: 74
1004	a[0]: ‘a’	1005	a[1]: ‘b’	1006	a[2]: ‘c’	1007	a[3]: ‘\0’
1008	ptr: 1001	1009	…	1010		1011	

Computer Memory Revisited

- Altering the value of a variable is indeed changing the content of the memory
  - e.g. i = 40; a[2] = ‘z’;

Addr	Content	Addr	Content	Addr	Content	Addr	Content
1000	i: 40	1001	j: 46	1002	k: 58	1003	m: 74
1004	a[0]: ‘a’	1005	a[1]: ‘b’	1006	a[2]: ‘z’	1007	a[3]: ‘\0’
1008	ptr: 1001	1009	…	1010		1011	

Addressing Concept

- Pointer stores the **address** of another entity
- It **refers** to a memory location

```
int i = 5;
int *ptr;                          /* declare a pointer variable */
ptr = &i;                          /* store address-of i to ptr */
printf("*ptr = %d\n", *ptr);       /* refer to referee of ptr */
```

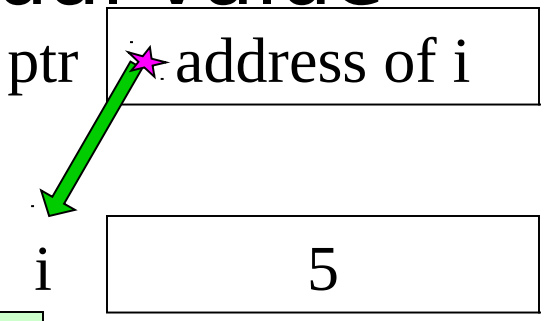
Why do we need Pointer?

- Simply because it’s there!
- It is used in some circumstances in C

What actually ***ptr*** is?

- **ptr** is a variable storing **an address**
- ptr is **NOT** storing the actual value of i

```
int i = 5;
int *ptr;
ptr = &i;
printf("i = %d\n", i);
printf("*ptr = %d\n", *ptr);
printf("ptr = %p\n", ptr);
```



Output:

```
i = 5
*ptr = 5
ptr = effff5e0
```

value of ptr = address of i in memory

Twin Operators

- &: Address-of operator
  - Get the *address* of an entity
    - e.g. **`ptr = &j;`**

Addr	Content	Addr	Content	Addr	Content	Addr	Content
1000	i: 40	1001	j: 33	1002	k: 58	1003	m: 74
1004	ptr: 1001	1005		1006		1007	

Twin Operators

- *: De-reference operator
  - Refer to the *content* of the referee
    - e.g. **`*ptr = 99;`**

Addr	Content	Addr	Content	Addr	Content	Addr	Content
1000	i: 40	1001	j: 99	1002	k: 58	1003	m: 74
1004	ptr: 1001	1005		1006		1007	

Example: Pass by Reference

- Modify behaviour in argument passing

An Illustration

```
int i = 5, j = 10;
int *ptr;
int **pptr;
ptr = &i;
pptr = &ptr;
*ptr = 3;
**pptr = 7;
ptr = &j;
**pptr = 9;
*pptr = &i;
*ptr = -2;
```

Data Table			
Name	Type	Description	Value
i	int	integer variable	5
j	int	integer variable	10

An Illustration

```
int i = 5, j = 10;
int *ptr;    /* declare a pointer-to-integer variable */
int **pptr;
ptr = &i;
pptr = &ptr;
*ptr = 3;
**pptr = 7;
ptr = &j;
**pptr = 9;
*pptr = &i;
*ptr = -2;
```

Data Table			
Name	Type	Description	Value
i	int	integer variable	5
j	int	integer variable	10
ptr	int *	integer pointer variable	

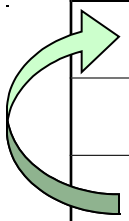
An Illustration

```
int i = 5, j = 10;
int *ptr;
int **pptr;  /* declare a pointer-to-pointer-to-integer variable */
ptr = &i;
pptr = &ptr;
*ptr = 3;
**pptr = 7;
ptr = &j;
**pptr = 9;
*pptr = &i;
*ptr = -2;
```

Data Table			
Name	Type	Description	Value
i	int	integer variable	5
j	int	integer variable	10
ptr	int *	integer pointer variable	
pptr	int **	integer pointer pointer variable	
		Double Indirection	

An Illustration

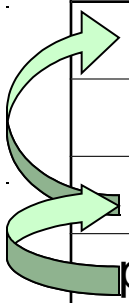
```
int i = 5, j = 10;
int *ptr;
int **pptr;
ptr = &i;     /* store address-of i to ptr */
pptr = &ptr;
*ptr = 3;
**pptr = 7;
ptr = &j;
**pptr = 9;
*pptr = &i;
*ptr = -2;
```



Data Table			
Name	Type	Description	Value
i	int	integer variable	5
j	int	integer variable	10
ptr	int *	integer pointer variable	address of i
pptr	int **	integer pointer pointer variable	
*ptr	int	de-reference of ptr	5

An Illustration

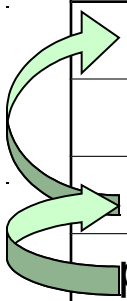
```
int i = 5, j = 10;
int *ptr;
int **pptr;
ptr = &i;
pptr = &ptr; /* store address-of ptr to pptr */
*ptr = 3;
**pptr = 7;
ptr = &j;
**pptr = 9;
*pptr = &i;
*ptr = -2;
```



Data Table			
Name	Type	Description	Value
i	int	integer variable	5
j	int	integer variable	10
ptr	int *	integer pointer variable	address of i
pptr	int **	integer pointer pointer variable	address of ptr
*pptr	int *	de-reference of pptr	value of ptr (address of

An Illustration

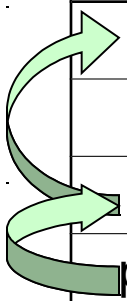
```
int i = 5, j = 10;
int *ptr;
int **pptr;
ptr = &i;
pptr = &ptr;
*ptr = 3;
**pptr = 7;
ptr = &j;
**pptr = 9;
*pptr = &i;
*ptr = -2;
```



Data Table			
Name	Type	Description	Value
i	int	integer variable	3
j	int	integer variable	10
ptr	int *	integer pointer variable	address of i
pptr	int **	integer pointer pointer variable	address of ptr
*ptr	int	de-reference of ptr	3

An Illustration

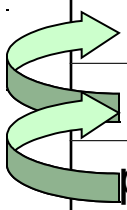
```
int i = 5, j = 10;
int *ptr;
int **pptr;
ptr = &i;
pptr = &ptr;
*ptr = 3;
**pptr = 7;
ptr = &j;
**pptr = 9;
*pptr = &i;
*ptr = -2;
```



Data Table			
Name	Type	Description	Value
i	int	integer variable	7
j	int	integer variable	10
ptr	int *	integer pointer variable	address of i
pptr	int **	integer pointer pointer variable	address of ptr
**pptr	int	de-reference of de-reference of pptr	7

An Illustration

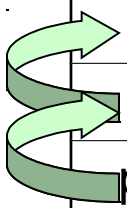
```
int i = 5, j = 10;
int *ptr;
int **pptr;
ptr = &i;
pptr = &ptr;
*ptr = 3;
**pptr = 7;
ptr = &j;
**pptr = 9;
*pptr = &i;
*ptr = -2;
```



Data Table			
Name	Type	Description	Value
i	int	integer variable	7
j	int	integer variable	10
ptr	int *	integer pointer variable	address of j
pptr	int **	integer pointer pointer variable	address of ptr
*ptr	int	de-reference of ptr	10

An Illustration

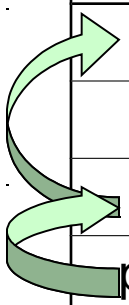
```
int i = 5, j = 10;
int *ptr;
int **pptr;
ptr = &i;
pptr = &ptr;
*ptr = 3;
**pptr = 7;
ptr = &j;
**pptr = 9;
*pptr = &i;
*ptr = -2;
```



Data Table			
Name	Type	Description	Value
i	int	integer variable	7
j	int	integer variable	9
ptr	int *	integer pointer variable	address of j
pptr	int **	integer pointer pointer variable	address of ptr
**pptr	int	de-reference of de-reference of pptr	9

An Illustration

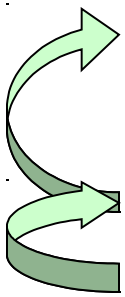
```
int i = 5, j = 10;
int *ptr;
int **pptr;
ptr = &i;
pptr = &ptr;
*ptr = 3;
**pptr = 7;
ptr = &j;
**pptr = 9;
*pptr = &i;
*ptr = -2;
```



Data Table			
Name	Type	Description	Value
i	int	integer variable	7
j	int	integer variable	9
ptr	int *	integer pointer variable	address of i
pptr	int **	integer pointer pointer variable	address of ptr
*pptr	int *	de-reference of pptr	value of ptr (address of

An Illustration

```
int i = 5, j = 10;
int *ptr;
int **pptr;
ptr = &i;
pptr = &ptr;
*ptr = 3;
**pptr = 7;
ptr = &j;
**pptr = 9;
*pptr = &i;
*ptr = -2;
```



Data Table			
Name	Type	Description	Value
i	int	integer variable	-2
j	int	integer variable	9
ptr	int *	integer pointer variable	address of i
pptr	int **	integer pointer pointer variable	address of ptr
*ptr	int	de-reference of ptr	-2

Pointer Arithmetic

- What's **`ptr + 1`**?
  - → The next memory location!
- What's **`ptr - 1`**?
  - → The previous memory location!
- What's **`ptr * 2`** and **`ptr / 2`**?
  - → Invalid operations!!!

Summary

- A pointer stores the **address** (memory location) of another entity
- Address-of operator (&) **gets the address** of an entity
- De-reference operator (*) **makes a reference** to the referee of a pointer
- Pointer and array
- Pointer arithmetic