Day20.C

Data Structure Introduction

Young W. Lim

December 12, 2017

1 Self-Referential Structure

1.1 Examples of declaring self-referential structure variables

```
::::::::::::::
h1.c
::::::::::::::
#include <stdio.h>

//.........................
struct aaa {
  int data;
  struct aaa *next;
};

struct aaa vara;

//.........................
struct bbb {
  int data;
  struct bbb *next;
} varb;

//.........................
struct ccc {
  int data;
  struct ccc *next;
};

typedef struct ccc ctype;

ctype varc;

//.........................
typedef struct ddd {
  int data;
  struct ddd *next;
} dtype;


dtype vard;



int main(void) {
  vara.data = 111;
  vara.next = NULL;

  varb.data = 222;
```

```
  varb.next = NULL;

  varc.data = 333;
  varc.next = NULL;

  vard.data = 444;
  vard.next = NULL;


  printf("vara.data=%d vara.next=%p \n", vara.data, vara.next);
  printf("varb.data=%d varb.next=%p \n", varb.data, varb.next);
  printf("varc.data=%d varc.next=%p \n", varc.data, varc.next);
  printf("vard.data=%d vard.next=%p \n", vard.data, vard.next);

}
::::::::::::::
h1.out
::::::::::::::
vara.data=111 vara.next=(nil)
varb.data=222 varb.next=(nil)
varc.data=333 varc.next=(nil)
vard.data=444 vard.next=(nil)

::::::::::::::
h2.c
::::::::::::::
#include <stdio.h>

//.........................
struct aaa vara;

struct aaa {
  int data;
  struct aaa *next;
};

//.........................
struct bbb {
  int data;
  struct bbb *next;
} varb;

//.........................
typedef struct ccc ctype;

ctype varc;

struct ccc {
  int data;
  struct ccc *next;
```

```
};


//.............................
typedef struct ddd {
  int data;
  struct ddd *next;
} dtype;


dtype vard;



int main(void) {
  vara.data = 111;
  vara.next = NULL;

  varb.data = 222;
  varb.next = NULL;

  varc.data = 333;
  varc.next = NULL;

  vard.data = 444;
  vard.next = NULL;


  printf("vara.data=%d vara.next=%p \n", vara.data, vara.next);
  printf("varb.data=%d varb.next=%p \n", varb.data, varb.next);
  printf("varc.data=%d varc.next=%p \n", varc.data, varc.next);
  printf("vard.data=%d vard.next=%p \n", vard.data, vard.next);

}
::::::::::::::
h2.out
::::::::::::::
vara.data=111 vara.next=(nil)
varb.data=222 varb.next=(nil)
varc.data=333 varc.next=(nil)
vard.data=444 vard.next=(nil)

::::::::::::::
h3.c
::::::::::::::
#include <stdio.h>

//.............................
struct aaa {
  int data;
```

```
  struct aaa *next;
};

struct aaa vara;
//.........................
struct bbb {
  int data;
  struct bbb *next;
} varb;

//.........................
typedef struct ccc ctype;

struct ccc {
  int data;
  ctype *next;
};

ctype varc;
//.........................
typedef struct ddd {
  int data;
  struct ddd *next;
} dtype;

dtype vard;



int main(void) {
  vara.data = 111;
  vara.next = NULL;

  varb.data = 222;
  varb.next = NULL;

  varc.data = 333;
  varc.next = NULL;

  vard.data = 444;
  vard.next = NULL;


  printf("vara.data=%d vara.next=%p \n", vara.data, vara.next);
  printf("varb.data=%d varb.next=%p \n", varb.data, varb.next);
  printf("varc.data=%d varc.next=%p \n", varc.data, varc.next);
  printf("vard.data=%d vard.next=%p \n", vard.data, vard.next);

}
::::::::::::::
```

```
h3.out
::::::::::::::
vara.data=111 vara.next=(nil)
varb.data=222 varb.next=(nil)
varc.data=333 varc.next=(nil)
vard.data=444 vard.next=(nil)

::::::::::::::
h4.c
::::::::::::::
#include <stdio.h>

//.........................
struct atype {
  int data;
  struct atype *next;
};

struct atype vara;
//.........................
struct btype {
  int data;
  struct btype *next;
} varb;

//.........................
typedef struct ctype ctype;

struct ctype {
  int data;
  ctype *next;
};

ctype varc;
//.........................
typedef struct dtype {
  int data;
  struct dtype *next;
} dtype;

dtype vard;



int main(void) {
  vara.data = 111;
  vara.next = NULL;

  varb.data = 222;
  varb.next = NULL;
```

```
  varc.data = 333;
  varc.next = NULL;

  vard.data = 444;
  vard.next = NULL;


  printf("vara.data=%d vara.next=%p \n", vara.data, vara.next);
  printf("varb.data=%d varb.next=%p \n", varb.data, varb.next);
  printf("varc.data=%d varc.next=%p \n", varc.data, varc.next);
  printf("vard.data=%d vard.next=%p \n", vard.data, vard.next);

}
::::::::::::::
h4.out
::::::::::::::
vara.data=111 vara.next=(nil)
varb.data=222 varb.next=(nil)
varc.data=333 varc.next=(nil)
vard.data=444 vard.next=(nil)
```

- all variables are globally defined
- defined outside `main` function
- global variables `vara, varb, varc, vard`
- results may be different for local variables

case view I

case 1	case 2

`struct aaa {`
`  int data;`
`  struct aaa *next;`
`};`
 	
`struct aaa vara;`
	`struct aaa vara;`
	
	`struct aaa {`
	`  int data;`
	`  struct aaa *next;`
	`};`

`struct bbb {`
`  int data;`
`  struct bbb *next;`
`} varb;`
	`struct bbb {`
	`  int data;`
	`  struct bbb *next;`
	`} varb;`

`struct ccc {`
`  int data;`
`  struct ccc *next;`
`};`
 	
`typedef struct ccc ctype;`
 	
`ctype varc;`
	`typedef struct ccc ctype;`
	
	`ctype varc;`
	
	`struct ccc {`
	`  int data;`
	`  struct ccc *next;`
	`};`

`typedef struct ddd {`
`  int data;`
`  struct ddd *next;`
`} dtype;`
 	
`dtype vard;`
	`typedef struct ddd {`
	`  int data;`
	`  struct ddd *next;`
	`} dtype;`
	
	`dtype vard;`

case view II

case 3	case 4

`struct aaa {`
`  int data;`
`  struct aaa *next;`
`};`
 	
`struct aaa vara;`
	`struct atype {`
	`  int data;`
	`  struct atype *next;`
	`};`
	
	`struct atype vara;`

`struct bbb {`
`  int data;`
`  struct bbb *next;`
`} varb;`
	`struct btype {`
	`  int data;`
	`  struct btype *next;`
	`} varb;`

`typedef struct ccc ctype;`
 	
`struct ccc {`
`  int data;`
`  ctype *next;`
`};`
 	
`ctype varc;`
	`typedef struct ctype ctype;`
	
	`struct ctype {`
	`  int data;`
	`  ctype *next;`
	`};`
	
	`ctype varc;`

`typedef struct ddd {`
`  int data;`
`  struct ddd *next;`
`} dtype;`
 	
`dtype vard;`
	`typedef struct dtype {`
	`  int data;`
	`  struct dtype *next;`
	`} dtype;`
	
	`dtype vard;`

- **variable view 1. vara**

Case 1	Case 2

`struct aaa {`
`  int data;`
`  struct aaa *next;`
`};`
 	
`struct aaa vara;`
	`struct aaa vara;`
	
	`struct aaa {`
	`  int data;`
	`  struct aaa *next;`
	`};`

Case 3	Case 4

`struct aaa {`
`  int data;`
`  struct aaa *next;`
`};`
 	
`struct aaa vara;`
	`struct atype {`
	`  int data;`
	`  struct atype *next;`
	`};`
	
	`struct atype vara;`

	case 1	case 2	case 3	case 4
tag name	aaa	aaa	aaa	atype
order	definition declaration	declaration definition	definition declaration	definition declaration
struct type	struct aaa	struct aaa	struct aaa	struct atype
pointer type	struct aaa *	struct aaa *	struct aaa *	struct atype *
var type	struct aaa	struct aaa	struct aaa	struct atype

- **variable view 2. varb**

Case 1	Case 2

`struct bbb {`
`  int data;`
`  struct bbb *next;`
`} varb;`
	`struct bbb {`
	`  int data;`
	`  struct bbb *next;`
	`} varb;`

Case 3	Case 4

`struct bbb {`
`  int data;`
`  struct bbb *next;`
`} varb;`
	`struct btype {`
	`  int data;`
	`  struct btype *next;`
	`} varb;`

	case 1	case 2	case 3	case 4
tag name	bbb	bbb	bbb	btype
order	simultaneous definition declaration	simultaneous definition declaration	simultaneous definition declaration	simultaneous definition declaration
struct type	struct bbb	struct bbb	struct bbb	struct btype
pointer type	struct bbb *	struct bbb *	struct bbb *	struct btype *
var type	struct bbb	struct bbb	struct bbb	struct btype

- **variable view 3. varc**

Case 1	Case 2

`struct ccc {`
`  int data;`
`  struct ccc *next;`
`};`
 	
`typedef struct ccc ctype;`
 	
`ctype varc;`
	`typedef struct ccc ctype;`
	
	`ctype varc;`
	
	`struct ccc {`
	`  int data;`
	`  struct ccc *next;`
	`};`

Case 3	Case 4

`typedef struct ccc ctype;`
 	
`struct ccc {`
`  int data;`
`  ctype *next;`
`};`
 	
`ctype varc;`
	`typedef struct ctype ctype;`
	
	`struct ctype {`
	`  int data;`
	`  ctype *next;`
	`};`
	
	`ctype varc;`

	case 1	case 2	case 3	case 4
tag name	ccc	ccc	ccc	ctype
order	definition typedef declaration	typedef declaration definition	typedef definition declaration	typedef definition declaration
struct type	struct ccc	struct ccc	struct ccc	struct ctype
pointer type	struct ccc *	struct ccc *	struct ccc *	struct ctype *
var type	ctype	ctype	ctype	ctype

- **variable view 4. vard**

Case 1	Case 2

`typedef struct ddd {`
`  int data;`
`  struct ddd *next;`
`} dtype;`
 	
`dtype vard;`
	`typedef struct ddd {`
	`  int data;`
	`  struct ddd *next;`
	`} dtype;`
	
	`dtype vard;`

Case 3	Case 4

`typedef struct ddd {`
`  int data;`
`  struct ddd *next;`
`} dtype;`
 	
`dtype vard;`
	`typedef struct dtype {`
	`  int data;`
	`  struct dtype *next;`
	`} dtype;`
	
	`dtype vard;`

	case 1	case 2	case 3	case 4
tag name	ddd	ddd	ddd	dtype
order	simultaneous typedef definition declaration	simultaneous typedef definition declaration	simultaneous typedef definition declaration	simultaneous typedef definition declaration
struct type	struct ddd	struct ddd	struct ddd	struct dtype
pointer type	struct ddd *	struct ddd *	struct ddd *	struct dtype *
var type	dtype	dtype	dtype	dtype

- **variable view summary (vara, varb, varc, vard)**

	case 1	case 2	case 3	case 4
tag name	aaa	aaa	aaa	atype
order	definition declaration	declaration definition	definition declaration	definition declaration
struct type	struct aaa	struct aaa	struct aaa	struct atype
pointer type	struct aaa *	struct aaa *	struct aaa *	struct atype *
var type	struct aaa	struct aaa	struct aaa	struct atype

	case 1	case 2	case 3	case 4
tag name	bbb	bbb	bbb	btype
order	simultaneous definition declaration	simultaneous definition declaration	simultaneous definition declaration	simultaneous definition declaration
struct type	struct bbb	struct bbb	struct bbb	struct btype
pointer type	struct bbb *	struct bbb *	struct bbb *	struct btype *
var type	struct bbb	struct bbb	struct bbb	struct btype

	case 1	case 2	case 3	case 4
tag name	ccc	ccc	ccc	ctype
order	definition typedef declaration	typedef declaration definition	typedef definition declaration	typedef definition declaration
struct type	struct ccc	struct ccc	struct ccc	struct ctype
pointer type	struct ccc *	struct ccc *	struct ccc *	struct ctype *
var type	ctype	ctype	ctype	ctype

	case 1	case 2	case 3	case 4
tag name	ddd	ddd	ddd	dtype
order	simultaneous typedef definition declaration	simultaneous typedef definition declaration	simultaneous typedef definition declaration	simultaneous typedef definition declaration
struct type	struct ddd	struct ddd	struct ddd	struct dtype
pointer type	struct ddd *	struct ddd *	struct ddd *	struct dtype *
var type	dtype	dtype	dtype	dtype

2 Linked List Data Structure

2.1 Case 1

```
//.............................
typedef struct node node;

struct node {
  int data;
  node *next;
};
//.............................
```

node.h

- a self-referential struture type definition
- a data member
- a next member (the address of the next node)

```
#include <stdio.h>
#include <stdlib.h>
#include "node.h"


//-----------------------------------
void init(node **sp, node **ep) {
  int   i;
  node *p;

  i=0;
  do {
    p = malloc( sizeof(node) );

    p->data = 10 * i++;
    p->next = NULL;

    if (*sp == NULL) *sp = p;
    if (*ep == NULL) *ep = p;
    else {
      (*ep)->next = p;
      (*ep) = p;
    }
  } while (i<10);
}
```

init.c

- initializes a linked list with 10 nodes
- allocates 10 nodes
- sets up the next member of each node
- `sp` points to the start node
- `ep` points to the end node

```
#include <stdio.h>
#include <stdlib.h>
#include "node.h"

//-----------------------------------
void display(node *sp) {
  node *p = sp;
  int i=0;

  while (p) {
    printf("%2d %3d \n", i++, p->data);
    p = p->next;
  }

}
```

display.c

- traverses each node in the linked list
- prints its data member

```
#include <stdio.h>
#include <stdlib.h>
#include "node.h"

//-----------------------------------
node *find(node *sp, int key) {
  node *p = sp;

  while (p) {
    if (p->data == key) return p;
    p = p->next;
  }

  return NULL;
}
```

find.c

- finds a node which has the same data member as the key
- returns the pointer to such a node

```
#include <stdio.h>
#include <stdlib.h>
#include "node.h"

//-----------------------------------
void insert(node *p, int key) {
  node *q, *r;

  r = malloc( sizeof(node) );

  q = p->next;

  p->next = r;
  r->next = q;
  r->data = key;

}
```

insert.c

- allocates a new node
- assigns the key to the data member of the newly allocated node
- inserts this node after the node which is pointed by `p`

```
#include <stdio.h>
#include <stdlib.h>
#include "node.h"

//-----------------------------------
void delete(node *p) {
  node *q, *r;

  q = p->next;
  r = q->next;

  p->next = r;

  free( q );
}
```

delete.c

- deletes the next node after the node which is pointed by `p`

```
#include <stdio.h>
#include <stdlib.h>
#include "node.h"

//-----------------------------------
void init(node **sp, node **ep);
node *find(node *sp, int key);
void display(node *sp);
void insert(node *p, int key);
void delete(node *p);


//-----------------------------------
int main(void) {
  node *p = NULL;   // current pointer
  node *sp= NULL;   // start pointer
  node *ep= NULL;   // end pointer

  init( &sp, &ep );

  p = find( sp, 50);

  printf("p->data = %d \n", p->data);

  insert( p, 55 );

  printf("\n* after inserting one node ...\n");
  display( sp );

  delete( p );

  printf("\n* after deleting one node ...\n");
  display( sp );

}
```

t3.c

- `p` points to the node whose data member is 50
- inserts a new node whose data member is 55, after the found node
- deletes the next node after the node which is pointed by `p`

2.2 Case 2

```
//.............................
typedef struct node node;

struct node {
  int data;
  node *next;
};
//.............................
```

node.h

- a self-referential struture type definition
- a data member
- a next member (the address of the next node)

```
#include <stdio.h>
#include <stdlib.h>
#include "node.h"


//-----------------------------------
void init(node **sp, node **ep) {
  int   i;
  node *p;

  i=0;
  do {
    p = malloc( sizeof(node) );

    p->data = 10 * i++;
    p->next = NULL;

    if (*sp == NULL) *sp = p;
    if (*ep == NULL) *ep = p;
    else {
      (*ep)->next = p;
      (*ep) = p;
    }
  } while (i<10);
}
```

init.c

- initializes a linked list with 10 nodes
- allocates 10 nodes
- sets up the next member of each node
- `sp` points to the start node
- `ep` points to the end node

```
#include <stdio.h>
#include <stdlib.h>
#include "node.h"

//-----------------------------------
void display(node *sp) {
  node *p = sp;
  int i=0;

  while (p) {
    printf("%2d %3d \n", i++, p->data);
    p = p->next;
  }

}
```

display.c

- traverses each node in the linked list
- prints its data member

```
#include <stdio.h>
#include <stdlib.h>
#include "node.h"

//-----------------------------------
node *find2(node *sp, int key, node **pp) {
  node *p = sp;

  while (p) {
    if (p->data == key) return p;
    *pp = p;
    p = p->next;
  }

  return NULL;
}
```

find2.c

- finds a node which has the same data member as the key
- returns the pointer to such a node
- pp is the pointer to the previous node

```
#include <stdio.h>
#include <stdlib.h>
#include "node.h"

//-----------------------------------
void insert2(node *p, node *pp, int key) {
  node *r;

  r = malloc( sizeof(node) );


  pp->next = r;
  r->next = p;
  r->data = key;

}
```

insert2.c

- allocates a new node
- assigns the key to the data member of the newly allocated node
- inserts this node before the node which is pointed by `p`
- after the node which is pointed by `pp`

```
#include <stdio.h>
#include <stdlib.h>
#include "node.h"

//-----------------------------------
void delete2(node *p, node *pp) {

  pp->next = p->next;

  free( p );
}
```

delete2.c

- deletes the node which is pointed by `p`

```
#include <stdio.h>
#include <stdlib.h>
#include "node.h"

//-----------------------------------
void init(node **sp, node **ep);
node *find(node *sp, int key);
void display(node *sp);
void insert(node *p, int key);
void delete(node *p);


//-----------------------------------
int main(void) {
  node *p = NULL;   // current pointer
  node *sp= NULL;   // start pointer
  node *ep= NULL;   // end pointer

  init( &sp, &ep );

  p = find( sp, 50);

  printf("p->data = %d \n", p->data);

  insert( p, 55 );

  printf("\n* after inserting one node ...\n");
  display( sp );

  delete( p );

  printf("\n* after deleting one node ...\n");
  display( sp );

}
```

t4.c

- `p` points to the node whose data member is 50
- inserts a new node whose data member is 55, before the found node
- deletes the node which is pointed by `p`