C Programming
Day18.B

2017.11.28

4 structure definitions

```c
#include <stdio.h>

struct aaa {
  int a;
  int b;
};

struct bbb {
  double a;
  double b;
};



void pr_aaa( struct aaa X, char *s ) {
  printf("--------------------\n");
  printf("%s.a = %d \n", s, X.a);
  printf("%s.b = %d \n", s, X.b);
}


void pr_bbb( struct bbb X, char *s ) {
  printf("--------------------\n");
  printf("%s.a = %g \n", s, X.a);
  printf("%s.b = %g \n", s, X.b);
}

int main(void) {


  struct aaa A = { 100,  200  };
  struct bbb B = { 11.1, 22.2 };
  struct bbb C;

  C = B;

  pr_aaa( A, "A");
  pr_bbb( B, "B");
  pr_bbb( C, "C");

}
```

```
--------------------
A.a = 100
A.b = 200
--------------------
B.a = 11.1
B.b = 22.2
--------------------
C.a = 11.1
C.b = 22.2
```

```c
#include <stdio.h>

struct bbb {
  double a;
  double b;
};

struct bbb ADD_bbb( struct bbb X, struct bbb Y) {
  struct bbb S;

  S.a = X.a + Y.a;
  S.b = X.b + Y.b;
  return(S);
}


int main(void) {
  struct bbb B = { 11.1, 22.2 };
  struct bbb C;

  C = B;

  printf("\nADD_bbb( B, C )============\n\n");
  printf("B.a= %f \n", B.a);
  printf("B.b= %f \n\n", B.b);

  printf("C.a= %f \n", C.a);
  printf("C.b= %f \n\n", C.b);

  C = ADD_bbb( B, C );

  printf("C.a= %f \n", C.a);
  printf("C.b= %f \n", C.b);

}
```

Structure Returning function

```
ADD_bbb( B, C )============

B.a= 11.100000
B.b= 22.200000

C.a= 11.100000
C.b= 22.200000

C.a= 22.200000
C.b= 44.400000
```

```c
#include <stdio.h>

struct bbb {
  double a;
  double b;
};

typedef struct bbb BT;

BT ADD_bbb( BT X, BT Y) {
  BT S;

  S.a = X.a + Y.a;
  S.b = X.b + Y.b;
  return(S);
}


int main(void) {
  BT B = { 11.1, 22.2 };
  BT C;

  C = B;

  printf("\nADD_bbb( B, C )============\n\n");
  printf("B.a= %f \n", B.a);
  printf("B.b= %f \n\n", B.b);

  printf("C.a= %f \n", C.a);
  printf("C.b= %f \n\n", C.b);

  C = ADD_bbb( B, C );

  printf("C.a= %f \n", C.a);
  printf("C.b= %f \n", C.b);


}
```

```c
#include <stdio.h>

struct bbb {
  double a;
  double b;
} B = {11.1,22.2}, C;

typedef struct bbb BT;

BT ADD_bbb( BT X, BT Y) {
  BT S;

  S.a = X.a + Y.a;
  S.b = X.b + Y.b;
  return(S);
}


int main(void) {

  C = B;

  printf("\nADD_bbb( B, C )============\n\n");
  printf("B.a= %f \n", B.a);
  printf("B.b= %f \n\n", B.b);

  printf("C.a= %f \n", C.a);
  printf("C.b= %f \n\n", C.b);

  C = ADD_bbb( B, C );

  printf("C.a= %f \n", C.a);
  printf("C.b= %f \n", C.b);


}
```

global variables

```
typedef struct bbb { double a; double b; }
        BT;
```

```c
#include <stdio.h>

typedef struct bbb {
    double a;
    double b;
} BT;


BT ADD_bbb( BT X, BT Y) {
  BT S;

  S.a = X.a + Y.a;
  S.b = X.b + Y.b;
  return(S);
}


int main(void) {
  BT B = {11.1,22.2}, C;

  C = B;

  printf("\nADD_bbb( B, C )============\n\n");
  printf("B.a= %f \n", B.a);
  printf("B.b= %f \n\n", B.b);

  printf("C.a= %f \n", C.a);
  printf("C.b= %f \n\n", C.b);

  C = ADD_bbb( B, C );

  printf("C.a= %f \n", C.a);
  printf("C.b= %f \n", C.b);


}
```

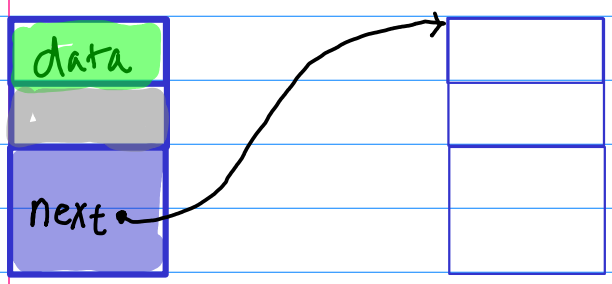
```c
#include <stdio.h>

struct aaa {
  int *addr;  // 8-byte
  int data;   // 4-byte
} var3;

typedef struct aaa atype;

int main(void) {
  struct aaa var1;
  atype      var2;

  int a = 100;

  var1.addr = &a;
  var1.data = a;

  printf("var1.addr = %p \n", var1.addr);
  printf("var1.data = %d \n", var1.data);

  printf("sizeof(var1) = %ld \n", sizeof(var1));

  var2.addr = var1.addr;
  var2.data = var1.data;

  printf("var2.addr = %p \n", var2.addr);
  printf("var2.data = %d \n", var2.data);

  printf("sizeof(var2) = %ld \n", sizeof(var2));

  var3.addr = var1.addr;
  var3.data = var1.data;

  printf("var3.addr = %p \n", var3.addr);
  printf("var3.data = %d \n", var3.data);

  printf("sizeof(var3) = %ld \n", sizeof(var3));
}
```

```
var1.addr = 0x7ffc5dc0626c
var1.data = 100
sizeof(var1) = 16
var2.addr = 0x7ffc5dc0626c
var2.data = 100
sizeof(var2) = 16
var3.addr = 0x7ffc5dc0626c
var3.data = 100
sizeof(var3) = 16
```

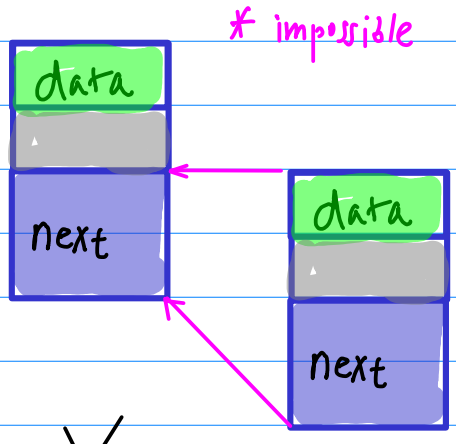
Self-Referential Structures

```c
#include <stdio.h>

struct aaa {
  int          data;   // 4-byte
  struct aaa *next;    // 8-byte
} ;

struct bbb {
  int          data;   // 4-byte
  // struct bbb  next;   // 8-byte  // Not Working
} ;


int main(void) {
  struct aaa var1;

  var1.data = 111;
  var1.next = NULL;

  printf("var1.next = %p \n", var1.next);
  printf("var1.data = %d \n", var1.data);

  printf("sizeof(var1) = %ld \n", sizeof(var1));

}
```

```
var1.next = (nil)
var1.data = 111
sizeof(var1) = 16
```



struct aaa * next;



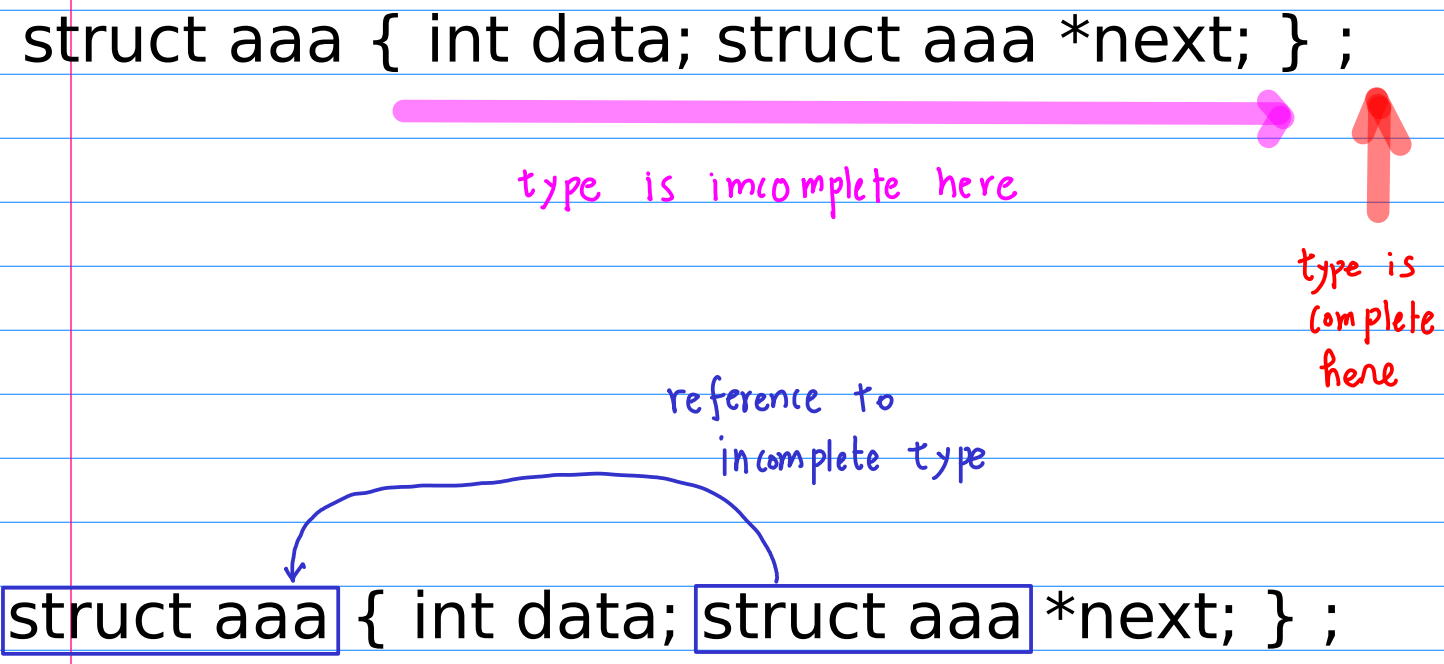
struct bbb ~~next~~ ;

A structure cannot contain
an instance of itself ordinary variables

A structure can contain
a reference to itself pointer variables

ordinary variables of a structure type
require its complete definition
(incomplete definition is not ok)

pointer variables of a structure type
does not require its complete definition
(incomplete definition is ok)


struct aaa { int data; struct aaa *next; } ;
type is imcomplete here
type is complete here
reference to incomplete type
struct aaa { int data; struct aaa *next; } ;

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

Struct aaa is used before finishing its definition

the use of a type specifier of the synthetic classes
(strucuter-type-reference or union-type-reference)
without a preceding definition
in the same or enclosing scope
is allowed

before finshing its definition

* when the size of the structure is not required
* pointers to the structure struct aaa * next
* a typdef name as a synonym for the structure

typedef struct aaa Atype

The use of this kind of specifier introduces
an incomplete definition of the type and type tag
in the innermost block containing the use.

For this definition to be completed,
a structure-type-definition or union-type-definition
must appear later in the same scope

C Reference Manual
Harbison & Steele

```c
#include <stdio.h>

//.....................
struct aaa {
  struct bbb *B;
};

//.....................
struct bbb {
  struct aaa *A;
};



int main(void) {

  struct bbb; // necessary!

  //.....................
  struct aaa {
    int a;
    struct bbb *B;
  };

  //.....................
  struct bbb {
    int b;
    struct aaa *A;
  };

  struct aaa A;
  struct bbb B;

  A.B = &B;
  B.A = &A;

}
```

```c
#include <stdio.h>

struct aaa {
  int          data;   // 4-byte
  struct aaa *next;   // 8-byte
} ;


typedef struct node node;

struct node {
  int   data;   // 4-byte
  node *next;   // 8-byte
} ;

int main(void) {
  struct aaa var1;
  node var2;


  var1.data = 111;
  var1.next = NULL;

  printf("var1.next = %p \n", var1.next);
  printf("var1.data = %d \n", var1.data);

  printf("sizeof(var1) = %ld \n", sizeof(var1));

  var2.data = 111;
  var2.next = NULL;

  printf("var2.next = %p \n", var2.next);
  printf("var2.data = %d \n", var2.data);

  printf("sizeof(var2) = %ld \n", sizeof(var2));

}
```

```
var1.next = (nil)
var1.data = 111
sizeof(var1) = 16
var2.next = (nil)
var2.data = 111
sizeof(var2) = 16
```