

자료구조 보고서

Homework#3

학과 : 소프트웨어학과

학번 : 2016039040

이름 : 윤용진

1. 강의영상에서 수행한 다음 프로그램에 대해 결과를 보고서로 제출하시오.

ap1.c, ap2.c, p2-1.c, p2-2.c, size.c, struct.c, padding.c

8. 프로그램 보고서에 각 소스코드별 본인 과제의 GitHub Repository URL을 명시한다.

<https://github.com/uniz21/DataStructure-Homework-3>

ap1.c 소스코드

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  void main()
5  {
6      printf("[----- [윤용진] [2016039040] -----]\n\n");
7
8      int list[5];
9      // 정수형 포인터를 담은 배열 선언
10     int *plist[5] = {NULL,};
11
12     // plist[0]이 가르키는 힙 영역에 정수형 크기(4바이트)의 영역을 할당
13     plist[0] = (int *)malloc(sizeof(int));
14
15     list[0] = 1;
16     list[1] = 100;
17
18     *plist[0] = 200;
19
20     printf("value of list[0] = %d\n", list[0]);
21     // list[0]의 주소
22     printf("address of list[0] = %p\n", &list[0]);
23     // list 값
24     printf("value of list = %p\n", list);
25     // list 주소
26     printf("address of list (&list) = %p\n", &list);
27     // list[0]의 주소 == list의 값 == list의 주소
28
29     printf("-----\n\n");
30
31     printf("value of list[1] = %d\n", list[1]);
32     // list[1]의 주소
33     printf("address of list[0] = %p\n", &list[1]);
34     // list + 1 의 값
35     printf("value of list = %d\n", *(list + 1));
36     // list + 1 의 주소
37     printf("address of list+1 = %p\n", list+1);
38     // list[1]==&list[1]--*(list+1)--list+1
39     // list+1 은 정수형 으로 선언된 list의 주소 + 4바이트(sizeof(int))
40
41     // list+1 은 정수형 으로 선언된 list의 주소 + 4바이트(sizeof(int))
42
43     printf("-----\n\n");
44
45     printf("value of *plist[0] = %d\n", *plist[0]);
46     // plist[0]의 주소
47     printf("&plist[0] = %p\n", &plist[0]);
48     // plist의 주소
49     printf("&plist = %p\n", &plist);
50     // plist의 값
51     printf("plist = %p\n", plist);
52     // plist[0]이 가르키는 영역의 주소
53     printf("plist[0] = %p\n", plist[0]);
54     // plist[1]이 가르키는 영역의 주소==null
55     printf("plist[1] = %p\n", plist[1]);
56     // plist[2]이 가르키는 영역의 주소==null
57     printf("plist[2] = %p\n", plist[2]);
58     // plist[3]이 가르키는 영역의 주소==null
59     printf("plist[3] = %p\n", plist[3]);
60     // plist[4]이 가르키는 영역의 주소==null
61     printf("plist[4] = %p\n", plist[4]);
62
63     // plist[0]의 주소와 plist[0]이 가르키는 주소는 다르다.(&plist[0]!=plist[0])
64     // 선언된 배열의 이름은 해당 배열 첫번째 항의 주소를 가리키고 주소로 갖는다.
65
66     free(plist[0]);
67 }
```

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

void main()
{
    printf("[----- [윤용진] [2016039040] -----]\n\n");

    int list[5];
    // 정수형 포인터를 담은 배열 선언
    int *plist[5] = {NULL,};

    // plist[0]이 가르키는 힙 영역에 정수형 크기(4바이트)의 영역을 할당
    plist[0] = (int *)malloc(sizeof(int));

    list[0] = 1;
    list[1] = 100;

    *plist[0] = 200;

    printf("value of list[0] = %d\n", list[0]);
    // list[0]의 주소
    printf("address of list[0] = %p\n", &list[0]);
    // list 값
    printf("value of list = %p\n", list);
    // list 주소
```

```

printf("address of list (&list) = %p\n", &list);
// list[0]의 주소 == list의 값 == list의 주소

printf("-----\n\n");

printf("value of list[1]    = %d\n", list[1]);
// list[1]의 주소
printf("address of list[0]  = %p\n", &list[1]);
// list + 1 의 값
printf("value of list      = %d\n", *(list + 1));
// list + 1 의 주소
printf("address of list+1   = %p\n", list+1);
// list[1]==&list[1]==*(list+1)==list+1
// list+1 은 정수형 으로 선언된 list의 주소 + 4바이트(sizeof(int))

printf("-----\n\n");

printf("value of *plist[0]  = %d\n", *plist[0]);
// plist[0]의 주소
printf("&plist[0]           = %p\n", &plist[0]);
// plist의 주소
printf("&plist              = %p\n", &plist);
// plist의 값
printf("plist              = %p\n", plist);
// plist[0]이 가르키는 영역의 주소
printf("plist[0]           = %p\n", plist[0]);
// plist[1]이 가르키는 영역의 주소==null
printf("plist[1]           = %p\n", plist[1]);
// plist[2]이 가르키는 영역의 주소==null
printf("plist[2]           = %p\n", plist[2]);
// plist[3]이 가르키는 영역의 주소==null
printf("plist[3]           = %p\n", plist[3]);
// plist[4]이 가르키는 영역의 주소==null
printf("plist[4]           = %p\n", plist[4]);

// plist[0]의 주소와 plist[0]이 가르키는 주소는 다르다.(&plist[0]!=plist[0])
// 선언된 배열의 이름은 해당 배열 첫번째 항의 주소를 값과 주소로 갖는다.

free(plist[0]);
}

```

ap1.c 실행결과

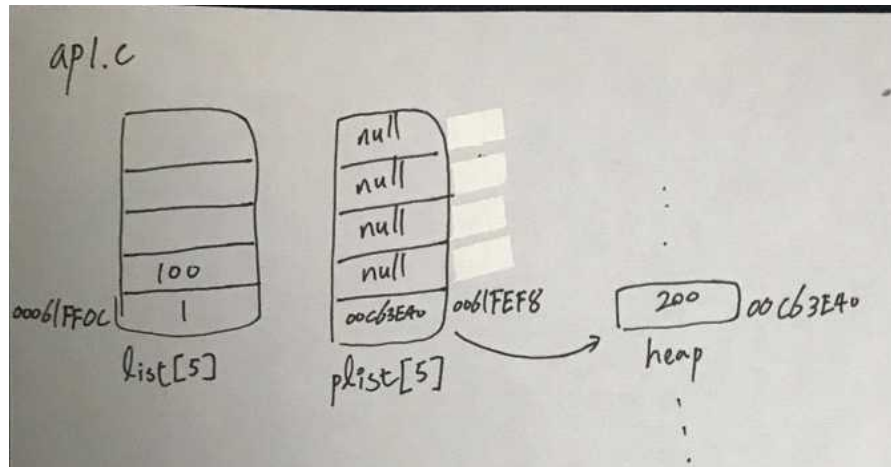
```
[Running] cd "c:\Users\yoony\OneDrive\
[----- [윤용진] [2016039040] -----]
```

```
value of list[0] = 1
address of list[0] = 0061FF0C
value of list = 0061FF0C
address of list (&list) = 0061FF0C
-----
```

```
value of list[1] = 100
address of list[0] = 0061FF10
value of list = 100
address of list+1 = 0061FF10
-----
```

```
value of *plist[0] = 200
&plist[0] = 0061FEF8
&plist = 0061FEF8
plist = 0061FEF8
plist[0] = 00C63E40
plist[1] = 00000000
plist[2] = 00000000
plist[3] = 00000000
plist[4] = 00000000
```

```
[Done] exited with code=1 in 1.948 sec
```



```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  void main()
5  {
6      printf("[----- [윤용진] [2016039040] -----]\n\n");
7
8      int list[5];
9      int *plist[5];
10
11     list[0] = 10;
12     list[1] = 11;
13
14     plist[0] = (int*)malloc(sizeof(int));
15
16     printf("list[0] \t= %d\n", list[0]);
17     printf("address of list \t= %p\n", list);
18     printf("address of list[0] \t= %p\n", &list[0]);
19     // list[0]의 주소 == list의 주소
20
21     // list + 0 의 주소 == list의 주소 + 0*4바이트
22     printf("address of list + 0 \t= %p\n", list+0);
23     // list + 1 의 주소 == list의 주소 + 1*4바이트
24     printf("address of list + 1 \t= %p\n", list+1);
25     // list + 2 의 주소 == list의 주소 + 2*4바이트
26     printf("address of list + 2 \t= %p\n", list+2);
27     // list + 3 의 주소 == list의 주소 + 3*4바이트
28     printf("address of list + 3 \t= %p\n", list+3);
29     // list + 4 의 주소 == list의 주소 + 4*4바이트
30     printf("address of list + 4 \t= %p\n", list+4);
31     // list[4]의 주소
32     printf("address of list[4] \t= %p\n", &list[4]);
33
34     // list + a == list[a]의 주소
35     // *(list + a) == list[a]의 값
36
37     free(plist[0]);
38 }

```

```

#include <stdio.h>
#include <stdlib.h>

```

```
void main()
```

```
{
```

```
    printf("[----- [윤용진] [2016039040] -----]\n\n");
```

```
    int list[5];
```

```
    int *plist[5];
```

```
    list[0] = 10;
```

```
    list[1] = 11;
```

```
    plist[0] = (int*)malloc(sizeof(int));
```

```
    printf("list[0] \t= %d\n", list[0]);
```

```

printf("address of list \t= %p\n", list);
printf("address of list[0] \t= %p\n", &list[0]);
// list[0]의 주소 == list의 주소

// list + 0 의 주소 == list의 주소 + 0*4바이트
printf("address of list + 0 \t= %p\n", list+0);
// list + 1 의 주소 == list의 주소 + 1*4바이트
printf("address of list + 1 \t= %p\n", list+1);
// list + 2 의 주소 == list의 주소 + 2*4바이트
printf("address of list + 2 \t= %p\n", list+2);
// list + 3 의 주소 == list의 주소 + 3*4바이트
printf("address of list + 3 \t= %p\n", list+3);
// list + 4 의 주소 == list의 주소 + 4*4바이트
printf("address of list + 4 \t= %p\n", list+4);
// list[4]의 주소
printf("address of list[4] \t= %p\n", &list[4]);

// list + a == list[a]의 주소
// *(list + a) == list[a]의 값

free(plist[0]);
}

```

ap2.c 실행결과

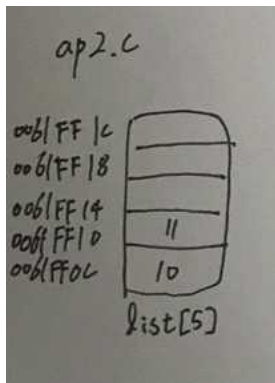
```

[Running] cd "c:\Users\yoony\OneDrive\
[----- [윤용진] [2016039040] -----]

list[0]    = 10
address of list      = 0061FF0C
address of list[0]   = 0061FF0C
address of list + 0   = 0061FF0C
address of list + 1   = 0061FF10
address of list + 2   = 0061FF14
address of list + 3   = 0061FF18
address of list + 4   = 0061FF1C
address of list[4]    = 0061FF1C

[Done] exited with code=1 in 2.15 sec

```



```

1  #include <stdio.h>
2
3  #define MAX_SIZE 100
4
5  float sum(float list[], int);
6  float input[MAX_SIZE], answer;
7  int i;
8
9  void main(void)
10 {
11     printf("[----- [윤용진] [2016039040] -----]\n\n");
12
13     for(i=0; i<MAX_SIZE; i++)
14         input[i] = i;
15
16     // input의 주소
17     printf("address of input = %p\n", input);
18
19     answer = sum(input, MAX_SIZE);
20     printf("The sum is : %f\n", answer);
21 }
22
23 float sum(float list[], int n)
24 {
25     // list의 값 == input의 주소
26     printf("value of list = %p\n", list);
27     // list의 주소 != input의 주소
28     printf("address of list = %p\n\n", &list);
29
30     /*
31     배열의 이름이 함수에 파라미터로 호출될 때 값이 복사되는 것이 아닌
32     주소를 받아 참조에 의해 호출된다
33     */
34
35     int i;
36     float tempsum = 0;
37     for(i = 0; i < n; i++)
38         tempsum += list[i];
39     return tempsum;
40 }

```

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

```
#define MAX_SIZE 100
```

```
float sum(float list[], int);
```

```
float input[MAX_SIZE], answer;
```

```
int i;
```

```
void main(void)
```

```
{
```

```
    printf("[----- [윤용진] [2016039040] -----]\n\n");
```

```
    for(i=0; i<MAX_SIZE; i++)
```



```

    input[i] = i;

// input의 주소
printf("address of input = %p\n", input);

answer = sum(input, MAX_SIZE);
printf("The sum is : %f\n", answer);
}

float sum(float list[], int n)
{
    // list의 값 == input의 주소
    printf("value of list = %p\n", list);
    // list의 주소 != input의 주소
    printf("address of list = %p\n\n", &list);

    /*
    배열의 이름이 함수에 파라미터로 호출될 때 값이 복사되는 것이 아닌
    주소를 받아 참조에 의해 호출된다
    */

    int i;
    float tempsum = 0;
    for(i = 0; i < n; i++)
        tempsum += list[i];
    return tempsum;
}

```

p2-1.c 실행결과

```

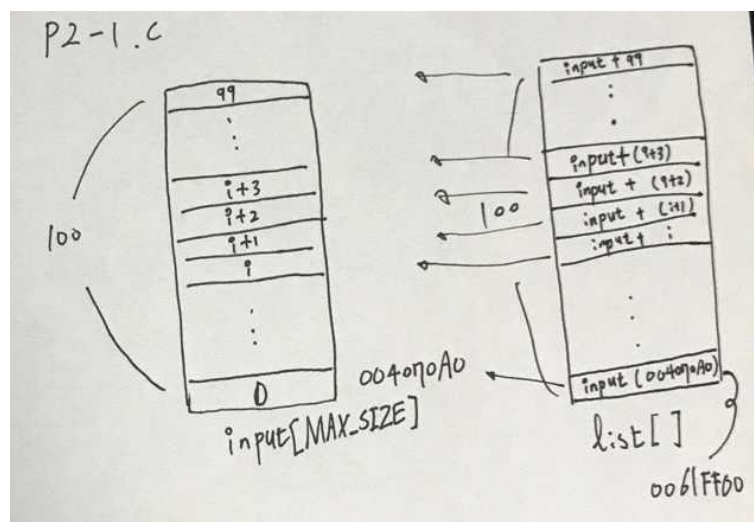
[Running] cd "c:\Users\yoony\OneDrive
[----- [윤용진] [2016039040] -----]

address of input = 004070A0
value of list = 004070A0
address of list = 0061FF00

The sum is : 4950.000000

[Done] exited with code=25 in 1.742 s

```




```

1  #include <stdio.h>
2
3  void print1 (int *ptr, int rows);
4
5  int main()
6  {
7      printf("[----- [윤용진] [2016039040] -----]\n\n");
8
9      int one[] = {0, 1, 2, 3, 4};
10
11     // 배열 one의 주소
12
13     printf("one      = %p\n", one);
14     printf("&one    = %p\n", &one);
15     printf("&one[0] = %p\n", &one[0]);
16     printf("\n");
17
18     // 배열 인덱스별 주소와 값 출력
19     print1(&one[0], 5);
20     // 정수형 배열인 one의 각 인덱스별 주소가 4바이트씩 증가
21
22     return 0;
23 }
24
25 void print1 (int *ptr, int rows)
26 {
27     int i;
28     printf("Address \t Contents\n");
29     for (i=0; i<rows; i++)
30         printf("%p \t %5d\n", ptr+i, *(ptr+i));
31     printf("\n");
32 }

```

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

```
void print1 (int *ptr, int rows);
```

```
int main()
```

```
{
```

```
    printf("[----- [윤용진] [2016039040] -----]\n\n");
```

```
    int one[] = {0, 1, 2, 3, 4};
```

```
    // 배열 one의 주소
```

```
    printf("one      = %p\n", one);
```

```
    printf("&one    = %p\n", &one);
```

```
    printf("&one[0] = %p\n", &one[0]);
```

```
    printf("\n");
```

```
    // 배열 인덱스별 주소와 값 출력
```

```
    print1(&one[0], 5);
```

```

// 정수형 배열인 one의 각 인덱스별 주소가 4바이트씩 증가

return 0;
}

void print1 (int *ptr, int rows)
{
    int i;
    printf("Address \t Contents\n");
    for (i=0; i<rows; i++)
        printf("%p \t %5d\n", ptr+i, *(ptr+i));
    printf("\n");
}

```

p2-2.c 실행결과

```

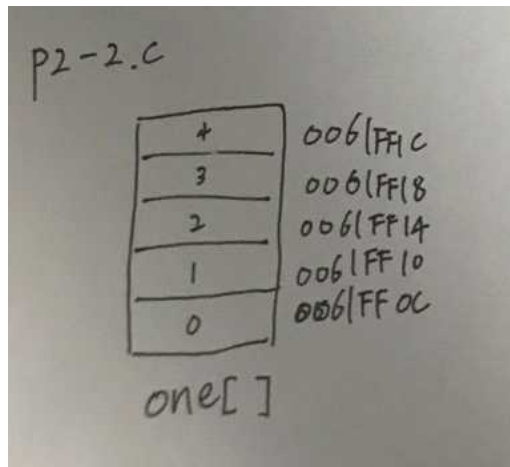
[Running] cd "c:\Users\yoony\OneDrive
[----- [문용진] [2016039040] -----]

one      = 0061FF0C
&one     = 0061FF0C
&one[0]  = 0061FF0C

Address      Contents
0061FF0C      0
0061FF10      1
0061FF14      2
0061FF18      3
0061FF1C      4

[Done] exited with code=0 in 1.668 se

```



size.c 소스코드

```
3주차 > C size.c > main()
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  void main()
5  {
6      printf("[----- [윤용진] [2016039040] -----]\n\n");
7
8      int **x;
9
10     // x의 크기 == *x의 주소의 크기(역참조 된 포인터)
11     printf("sizeof(x) = %lu\n", sizeof(x));
12     // *x의 크기 == **x의 주소의 크기(역참조 된 포인터)
13     printf("sizeof(*x) = %lu\n", sizeof(*x));
14     // **x의 크기 == **x의 크기(정수형)
15     printf("sizeof(**x) = %lu\n", sizeof(**x));
16
17 }
```

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

void main()
{
    printf("[----- [윤용진] [2016039040] -----]\n\n");

    int **x;

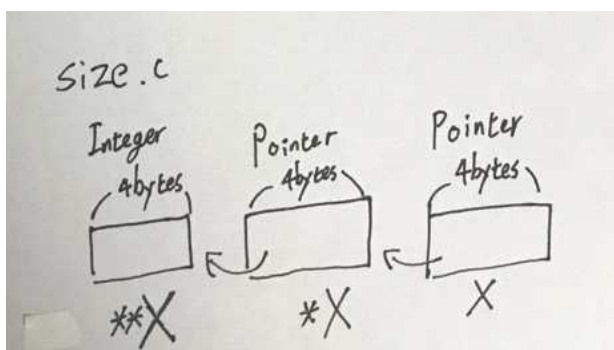
    // x의 크기 == *x의 주소의 크기(역참조 된 포인터)
    printf("sizeof(x) = %lu\n", sizeof(x));
    // *x의 크기 == **x의 주소의 크기(역참조 된 포인터)
    printf("sizeof(*x) = %lu\n", sizeof(*x));
    // **x의 크기 == **x의 크기(정수형)
    printf("sizeof(**x) = %lu\n", sizeof(**x));
}
```

size.c 실행결과

```
[Running] cd "c:\Users\yoony\OneDrive\
[----- [윤용진] [2016039040] -----]

sizeof(x) = 4
sizeof(*x) = 4
sizeof(**x) = 4

[Done] exited with code=16 in 2.818 se
```



struct.c 소스코드

```
1  #include <stdio.h>
2
3  struct student1 {
4      char lastName;
5      int studentId;
6      char grade;
7  };
8
9  typedef struct {
10     char lastName;
11     int studentId;
12     char grade;
13 } student2;
14
15 int main()
16 {
17     printf("[----- [윤용진] [2016039040] -----]\n\n");
18
19     //typedef로 선언한 구조체가 아니기 때문에 구조체임을 명시해야 한다.
20     struct student1 st1 = {'A',100,'A'};
21
22     printf("st1.lastName = %c\n", st1.lastName);
23     printf("st1.studentId = %d\n", st1.studentId);
24     printf("st1.grade = %c\n", st1.grade);
25
26     student2 st2 = {'B',200,'B'};
27
28     printf("st2.lastName = %c\n", st2.lastName);
29     printf("st2.studentId = %d\n", st2.studentId);
30     printf("st2.grade = %c\n", st2.grade);
31
32     student2 st3;
33
34     //구조치환(옛날 컴파일러는 불가능하다.)
35     st3 = st2;
36
37     printf("st3.lastName = %c\n", st3.lastName);
38     printf("st3.studentId = %d\n", st3.studentId);
39     printf("st3.grade = %c\n", st3.grade);
40
41     /*
42     //전체 구조의 동등성 검사(비교불가), 각각 비교해야한다.
43     if(st3 == st2)
44         printf("equal\n");
45     else
46         printf("not equal\n");
47     */
48     return 0;
49 }
```

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

```
struct student1 {
    char lastName;
    int studentId;
    char grade;
};
```

```
typedef struct {
    char lastName;
    int studentId;
    char grade;
} student2;
```

```
int main()
{
```

```

printf("[----- [윤용진] [2016039040] -----]\n\n");

//typedef로 선언한 구조체가 아니기 때문에 구조체임을 명시해야 한다.
struct student1 st1 = {'A',100,'A'};

printf("st1.lastName = %c\n", st1.lastName);
printf("st1.studentId = %d\n", st1.studentId);
printf("st1.grade = %c\n", st1.grade);

student2 st2 = {'B',200,'B'};

printf("st2.lastName = %c\n", st2.lastName);
printf("st2.studentId = %d\n", st2.studentId);
printf("st2.grade = %c\n", st2.grade);

student2 st3;

//구조치환(옛날 컴파일러는 불가능하다.)
st3 = st2;

printf("st3.lastName = %c\n", st3.lastName);
printf("st3.studentId = %d\n", st3.studentId);
printf("st3.grade = %c\n", st3.grade);

/*
//전체 구조의 동등성 검사(비교불가), 각각 비교해야한다.
if(st3 == st2)
    printf("equal\n");
else
    printf("not equal\n");
*/
return 0;
}

```

struct.c 실행결과

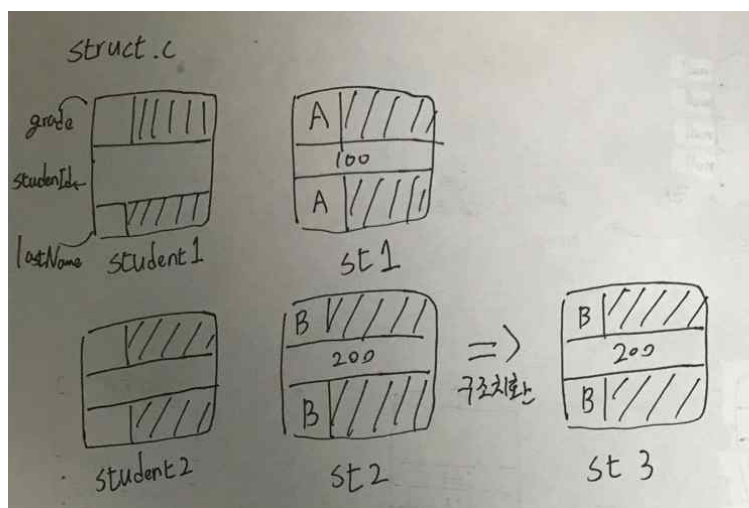
```

[Running] cd "c:\Users\yoony\OneDrive
[----- [윤용진] [2016039040] -----]

st1.lastName = A
st1.studentId = 100
st1.grade = A
st2.lastName = B
st2.studentId = 200
st2.grade = B
st3.lastName = B
st3.studentId = 200
st3.grade = B

[Done] exited with code=0 in 1.619 se

```



```

1  #include <stdio.h>
2
3  struct student {
4      char lastName[13]; /* 13bytes */ //padding 3bytes
5      int studentId; /* 4bytes */
6      short grade; /* 2bytes */ //padding 2bytes
7  };
8  // 구조체 student의 변수 크기 합 = 19bytes
9
10 int main()
11 {
12     printf("[----- [윤용진] [2016039040] -----]\n\n");
13
14     /**padding 처리는 Compiler마다 다르다.***/
15     /*
16     - 구조는 같은 메모리 경계에서 시작하고 끝나야 함
17     - 짝수바이트거나 4, 8, 16등의 배수가 되는 메모리 경계
18     */
19
20     struct student pst;
21
22     // 구조체 student의 크기 출력 = 24bytes(padding)
23     printf("size of student = %ld\n", sizeof(struct student));
24     printf("size of int = %ld\n", sizeof(int));
25     printf("size of short = %ld\n", sizeof(short));
26
27     return 0;
28 }

```

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

```

struct student {
    char lastName[13]; /* 13bytes */ //padding 3bytes
    int studentId; /* 4bytes */
    short grade; /* 2bytes */ //padding 2bytes
};
// 구조체 student의 변수 크기 합 = 19bytes

```

```

int main()
{
    printf("[----- [윤용진] [2016039040] -----]\n\n");

    /**padding 처리는 Compiler마다 다르다.***/
    /*
    - 구조는 같은 메모리 경계에서 시작하고 끝나야 함
    - 짝수바이트거나 4, 8, 16등의 배수가 되는 메모리 경계
    */

    struct student pst;

    // 구조체 student의 크기 출력 = 24bytes(padding)
    printf("size of student = %ld\n", sizeof(struct student));

```



```
printf("size of int = %ld\n", sizeof(int));  
printf("size of short = %ld\n", sizeof(short));  
  
return 0;  
}
```

padding.c 실행결과

```
[Running] cd "c:\Users\yoony\OneDrive  
[----- [문용진] [2016039040] -----]  
  
size of student = 24  
size of int = 4  
size of short = 2  
  
[Done] exited with code=0 in 2.232 s
```

