```
/*************************************************************/
// [1-3-3] : 문자열 길이 측정 함수
/*************************************************************/

unsigned int  str_lenth(const char * d)
{
   int cnt=0;

   while(*d++) cnt++;
   return cnt;
}

/*************************************************************/
// [1-3-4] : 문자열 연결 함수
/*************************************************************/

void str_add(char * d, const char * s)
{
   while(*d++);
   *d--;
   while(*d++=*s++);
}

/*************************************************************/
// [1-3-5] : 문자열 비교 함수
/*************************************************************/

int str_comp(const char *a, const char *b)
{
   do
   {
      if(*a<*b) return -1;
      if(*a>*b) return 1;
      b++;
   }while(*a++);

   return 0;
}


/*************************************************************/
// [1-3-8] : 단일 포인터
/*************************************************************/

void Swap(int *p, int *q)
{
   int t;

   t = *p;
   *p = *q;
   *q= t;
}

void main(void )
{
   Swap(&a, &b);
}

#endif

/*************************************************************/
// [1-3-9] : 다중 포인터의 활용
/*************************************************************/

int func(const int **p)
```

```
{
    int i;
    int sum = 0;

    printf("==========%d\n", (*p)[0]);

    for(i=1; i<= (*p)[0]; i++)
    {
       sum+=(*p)[i];
    }
     *p += i;

    return sum;
}

/************************************************************/
// [2-2-1] : 배열의 요소 타입
/************************************************************/

// 1

int (*b1[4])[4];
int (**b2[4])[4];
int b3[4];

// 2

int * b1[4];
int (*b2[4])[3];
int b3[4];

// 3

int ** b1[4];
int *(*b2[4])[3];
int * b3[4];

/************************************************************/
// [2-2-2] : 함수의 리턴 타입
/************************************************************/

// 1

int * f1(void)
int ** f2(void)
int f3(void)

// 2

int (*f1(void))[4]
int (**f2(void))[4]
int   f3(void)

// 3

int (*f1(void))(int)
int (**f2(void))(int)
int   f3(void)

/************************************************************/
// [2-3-3] : 포인터 배열
/************************************************************/

// 다음 2가지 모두 가능
```

```
*a[1]= 30;
a[3][2] = 30;

/************************************************************/
// [2-3-7] : 문자열 바꾸기
/************************************************************/

void Swap(char **p, char **q)
{
   char *t;

   t = *p;
   *p = *q;
   *q = t;
}

void main(void)
{
   Swap(&a, &b);
}

/************************************************************/
// [2-3-10] : 배열 포인터 연습
/************************************************************/

void func(int (*p)[4])
{
   (*p)[2]= 50;
}

/************************************************************/
// [2-3-13] : 구조체 주소의 함수 전달
/************************************************************/

// 다음 3가지 모두 가능

(*test).score = 100;
test->score = 100;
test[0].score = 100;

/************************************************************/
// [2-4-3] : 2차원배열의 전달
/************************************************************/

void draw_pixel(int y, int x, int value, int(*p)[3])

/************************************************************/
// [2-4-4] : 2차원배열의 리턴
/************************************************************/

int (*func(void))[4]

/************************************************************/
// [2-4-5] : 배열 등가포인터 연습
/************************************************************/

int (*pa)[3][4];
int (**pb)[4];
int *(**pc)(int *);
int *(*pd)[4];
int (*(**pe)(void))[4];

/************************************************************/
// [2-4-8] : 배열의 액세스
/************************************************************/
```

```
// 다음 3가지 모두 가능

printf("%d\n", b[0][3]);
printf("%d\n", b[1][7]);
printf("%d\n", b[2][11]);

/************************************************************/
// [2-4-9] : 환산법의 적용
/************************************************************/

printf("%d\n", p[11]);

/************************************************************/
// [2-4-10] : 함수 parameter의 직관적 설계
/************************************************************/

void f1( int *p)
{
   printf("%d\n", p[12 + 4 + 1]);
}

void f2(int(*p)[4])
{
   printf("%d\n", p[2][-11]);
}

void f3( int(*p)[3][4] )
{
   printf("%d\n", p[1][2][1]);
}

void f4( int(*p)[3][4] )
{
   printf("%d\n", (p-3)[1][2][1]);
}

void f5( int(*p)[2][3][4] )
{
   printf("%d\n", (*p)[1][2][1]);
}

void f6( int(*p)[2][3][4] )
{
   printf("%d\n", p[1][1][2][1]);
}

/************************************************************/
// [2-4-11] : 함수 리턴의 직관적 설계
/************************************************************/

int (*f1(void) )[3]
int (**f2(void))[3]
int *f3(void)
int (**f4(void))[3]
int (*(*f5(void))[2])[3]

void main(void)
{
   printf("6=%d\n",     f1()[1][2]);
   printf("6=%d\n",     (*f2())[1][2]);
   printf("6=%d\n",     (f3()+1)[5]);
   printf("6=%d\n",     f4()[1][1][2]);
   printf("6=%d\n",     (*f5())[1][1][2]);
}
```

```
/*************************************************************/
// [2-5-1] : 함수 등가포인터의 실행
/*************************************************************/

int (*p)(int a, int b);
void (*q)(void);
int * (*r)(void);

p = add;
q = f1;
r = f2;

printf("%d\n", p(3,4));
q();
printf("%d\n", r()[2]);

/*************************************************************/
// [2-5-2] : 함수를 함수에 전달하자
/*************************************************************/

void func( int (*p)(int a, int b) )

/*************************************************************/
// [2-5-3] : Parameter 미지정형
/*************************************************************/

int(*p)();

/*************************************************************/
// [2-5-5] : 함수 Lookup table
/*************************************************************/

int (*fa[3])(int a, int b) = {add, sub, mul};

/*************************************************************/
// [2-5-6] : 함수를 받고 함수를 리턴하는 함수
/*************************************************************/

int (*op(int(*fp)(void)))(int, int)

/*************************************************************/
// [2-5-8] : typedef을 이용한 가독성의 증대
/*************************************************************/

typedef int (*FPTR)();

/*************************************************************/
// [2-5-9] : 직관적 코드 설계 예제 1
/*************************************************************/

// 1
int (*f2(void))[4]
int (*(*f1(void))(void))[4]

printf("%d\n", f1( )( )[1][2]);

// 2

typedef int (*FP2)[4];
typedef FP2 (*FP1)(void);

printf("%d\n", f1( )( )[1][2]);

/*************************************************************/
```

```
// [2-5-10] : 직관적 코드 설계 예제 2
/*************************************************************/

int   (*f2(void))[4]
int *f1(void)

printf("%d\n", f1()[-2]);

/*************************************************************/
// [2-5-11] : 직관적 코드 설계 예제 3
/*************************************************************/

int (*f2(void))[3][4]
int(*f1(void))[4]

printf("%d\n", f1( )[1][2]);

/*************************************************************/
// [2-5-12] : 직관적 코드 설계 예제 4
/*************************************************************/

struct _st (*f2(void))[3]
struct _st * f1(int num)

printf("%s\n", f1(0)[4].name + 1);

/*************************************************************/
// [2-5-13] : 직관적 코드 설계 예제 5
/*************************************************************/

// 1

int *(**func1(void))()
int *(*func2(int(*p)(void)))()

printf("%d\n", func2(f4)()[3]);

// 2

typedef int *(*FPTR)();

FPTR fa[2] = {f1, f2};
FPTR* func1(void)
FPTR func2(int(*p)())

printf("%d\n", func2(f4)()[3]);

/*************************************************************/
// [2-8-1] : Type casting 연습 1
/*************************************************************/

// 1

printf("%f\n", ((double*)x)[0]);
printf("%f\n", ((double*)x)[1]);
printf("%f\n", ((double*)x)[2]);

// 2

for(i=0; i<3; i++)
{
   printf("%f\n", ((double*)x)[i]);
}

/*************************************************************/
```

```
// [2-8-2] : Type casting 연습 2
/*************************************************************/

printf("%f\n", (*(double**)p)[i]);

/*************************************************************/
// [2-8-3] :  Type casting 연습 3
/*************************************************************/

printf("%s\n", (*(char**)p));

/*************************************************************/
// [2-8-4] : Type casting 연습 4
/*************************************************************/

// 1

printf("%d\n", ((struct st*)&a)->i);
printf("%c\n", ((struct st*)&a)->c);

// 2

printf("%d\n", (*(struct st*)&a).i);
printf("%c\n", (*(struct st*)&a).c);

/*************************************************************/
// [2-8-5] : int 변수로 함수 실행하기
/*************************************************************/

printf("%d\n", ((int(*)(int,int))a)(3,4));

/*************************************************************/
// [2-8-6] : Type casting 연습 5
/*************************************************************/

printf("%d\n", (*(int(**)(int,int))p)(3,4));

/*************************************************************/
// [2-8-8] : 가변의 인수전달
/*************************************************************/

// 1

sum += *(&cnt + i);

// 2

sum += (&cnt)[i];

/*************************************************************/
// [2-8-9] : 가변인자의 access
/*************************************************************/

printf("%u\n", *(unsigned int *)(&a+1));
printf("%c\n", *(unsigned char *)(&a+2));
printf("%f\n", *(double *)(&a+3));
printf("%f\n", *(double *)(&a+5));

/*************************************************************/
// [1-8-11] : 실수의 메모리 내용 dump
/*************************************************************/

printf("float : %#.8x\n", *(unsigned int *)&a);
printf("double: %#.8x : %.8x\n\n", *((unsigned int *)&b+1), *(unsigned int *)&b);
```

```
/*************************************************************/
// [2-8-12] : 간이 format 지시자의 이용
/*************************************************************/

case 'u' : printf("%u\n", *(unsigned int *)ap++); break;
case 'd' : printf("%d\n", *ap++); break;
case 'c' : printf("%c\n", *(char *)ap++); break;
case 'f' : printf("%f\n", *(double *)ap); ap+=2;break;
case 's' : printf("%s\n", *(char **)ap++); break;

/*************************************************************/
// [2-8-14] : my_add 함수의 재설계
/*************************************************************/

int my_add(int cnt, ...)
{
   int i, sum = 0;

   va_list ap;
   va_start(ap, cnt);

   for(i = 1; i<= cnt; i++)
   {
      sum += va_arg(ap, int);
   }

   return sum;
}

/*************************************************************/
// [2-8-16] : Data Parsing 연습 1
/*************************************************************/

printf("%d\n", ((struct _st*)(&c+4))->i);
printf("%c\n", ((struct _st*)(&c+4))->c);
printf("%s\n", *(char**)(&c+12));
printf("%f\n", (*(double**)(&c+16))[0] );
printf("%f\n", (*(double**)(&c+16))[1] );
printf("%f\n", (*(double**)(&c+16))[2] );
printf("%d\n", (*(int(**)(int,int))(&c+20))(3,4));

/*************************************************************/
// [2-8-17] : Data Parsing 연습 2
/*************************************************************/

printf("f => %.2f\n",  **(double**)(&a+1));
printf("%s", (*(char*(***)())(&a+2))[0]()+2);      // (*(FPTR*)(&a+2))[0]()+2
printf("%s\n", (*(char*(***)())(&a+2))[1]()+2);    //(*(FPTR*)(&a+2))[1]()+2

/*************************************************************/
// [3-1-5] : 연속한 구조체 데이터
/*************************************************************/

printf("0x%x, 0x%x, 0x%x\n",(info+i)->x, (info+i)->y, (info+i)->length);

/*************************************************************/
// [3-1-6] : 연속한 구조체는 구조체 배열이다
/*************************************************************/

printf("0x%x, 0x%x, 0x%x\n",info[i].x, info[i].y, info[i].length);

/*************************************************************/
// [3-1-11] : pack으로 원하는 구조체 만들기
/*************************************************************/
```

```
#pragma pack(push, 2)

/*************************************************************/
// [3-2-3] : 비트필드 구조체의 활용
/*************************************************************/

unsigned short sec : 5;
unsigned short min : 6;
unsigned short hour : 5;

/*************************************************************/
// [3-2-8] : 공용체와 구조체
/*************************************************************/

union float_data
{
   float f;

   struct
   {
      unsigned int mant:23;
      unsigned int exp:8;
      unsigned int sign:1;
   }bit;
}fdata;

/*************************************************************/
// [3-2-9] : 파일시스템의 파일 종류 인식
/*************************************************************/

if(fn[i].v && fn[i].s && fn[i].h && fn[i].r) printf("Long File Name\n");

/*************************************************************/
// [3-2-10] : 효율적인 long file name의 판단
/*************************************************************/

if(fn[i].l.ln == 0xF) printf("Long File Name\n");
```