

학번 : 이름 :

[포인터]

※ 포인터 변수의 크기

포인터 변수	크기(byte)
int *pa	*pa(4 byte), pa(8 byte[64bit] / 4 byte[32bit])
double *pb	*pb(8 byte), pb(8 byte[64bit] / 4 byte[32bit])
char *pc	*pc(1 byte), pc(8 byte[64bit] / 4 byte[32bit])
long *pd	*pd(8 byte), pd(8 byte[64bit] / 4 byte[32bit])

※ 포인터 변수에는 데이터 타입 선언이 왜 필요할까?

포인터가 가르치는 주소에 있는 데이터를 어떤 자료형으로 해석할지 컴파일러에게 알려주기 위해 필요하다. (즉, 역참조 *p할 때 데이터를 몇 바이트 읽을지, p + 1 같은 포인터 연산에서 주소를 얼마나 이동할지 알기 위해)

※ 포인터 연산

연산자	연산이 갖는 의미
+, *, / 등	error: invalid operands to binary expression ('int *' and 'int *')
- (뺄셈)	두 포인터 사이의 상대적 거리
= 또는 != (관계 연산자)	두 포인터가 같은 주소를 가리키는지 비교
= (대입연산자)	한 포인터가 가진 주소값을 다른 포인터에 대입

학번 : 이름 :

[포인터와 문자 배열]

※ 아래 코드의 문제점을 서술하고 해결 방법을 제시해 보시오.

```
char *p;  
printf("문자열 입력: ");  
scanf("%s", p);  
printf("문자열 출력: %s\n", p);
```

문제점: p가 가리킬 메모리 공간을 확보하지 않은 상태에서 입력을 저장하려고 함

해결방법 1:

```
char p[100];  
printf("문자열 입력: ");  
scanf("%99s", p);  
printf("문자열 출력: %s\n", p);
```

해결방법 2:

```
char *p = malloc(100);  
  
printf("문자열 입력: ");  
scanf("%99s", p);  
printf("문자열 출력: %s\n", p);  
  
free(p);
```

1. 실습예제1- ex01.c

- 코드와 출력 결과 붙여 넣기

코드	출력결과
#include <stdio.h>	문자열 입력: info 문자열 출력: info

학번 : 이름 :

<pre>int main(void) { char *p; char str[100]; p = str; printf("문자열 입력: "); scanf("%99s", p); printf("문자열 출력: %s\n", p); return 0; }</pre>	
---	--

2. 실습예제2- ex02.c

- 코드와 출력 결과 붙여 넣기

코드	출력결과
<pre>#include <stdio.h> int main(void) { char c[] = "computer"; char *p = c; while (*p != '\0') { printf("%c", *p); p++; } printf("\n"); return 0; }</pre>	computer

3. 실습예제3- ex03.c

- 코드와 출력 결과 붙여 넣기

학번 : 이름 :

코드	출력결과
<pre>#include <stdio.h> int main(void) { char *p = "computer"; printf("%s\n", p); return 0; }</pre>	computer

학번 : 이름 :

[포인터와 숫자 배열]

※ 배열의 이름은 (포인터)처럼 동작한다.

※ 포인터를 배열을 가리키게 하면 배열의 (첫 번째 원소의 주소)을(를) 가리킨다.

※ `int a[]={1,2,3,4,5}`이고, 메모리구조가 아래의 그림과 같을 때 표의 빈 칸을 채워보시오.

0x0012ff60	1
0x0012ff61	
0x0012ff62	
0x0012ff63	
0x0012ff64	2
0x0012ff65	
0x0012ff66	
0x0012ff67	
0x0012ff68	...

<code>&a</code>	0x0012ff60
<code>&a[0]</code>	0x0012ff60
<code>a[0]</code>	1
<code>&a[1]</code>	0x0012ff64
<code>a[1]</code>	2

[포인터와 2차원 배열]

※ `int arr[3][2]={ {1,2}, {3,4}, {5,6} }`에서 1, 4, 6을 포인터를 이용하여 출력해 보시오.

```
#include <stdio.h>

int main(void) {
    int arr[3][2] = {
        {1, 2},
```

학번 : 이름 :

```
    {3, 4},
    {5, 6}
};

printf("%d\n", (*(arr + 0) + 0)); // 1
printf("%d\n", (*(arr + 1) + 1)); // 4
printf("%d\n", (*(arr + 2) + 1)); // 6

return 0;
}
```

※char str[3][5]={{"AB"}, {"CDEF"}, {"GHIJ"}}에서 AB, CDEF, IJ, H를 포인터를 이용하여 각각 출력해 보세요.

```
#include <stdio.h>

int main(void) {
    char str[3][5] = {
        {"AB"},
        {"CDEF"},
        {"GHIJ"}
    };

    printf("%s\n", *(str + 0));           // AB
    printf("%s\n", *(str + 1));           // CDEF
    printf("%s\n", *(str + 2) + 2);       // IJ
    printf("%c\n", (*(str + 2) + 1));     // H

    return 0;
}
```

4. 실습예제4- ex04.c

- 코드와 출력 결과 붙여 넣기

코드	출력결과
<pre>#include <stdio.h> #include <string.h> int main(void) { char city[5][7] = { "Seoul",</pre>	[도시이름] Seoul Pusan Incheon Suwon Daegu [n으로 끝나는 도시만 출력] Pusan Incheon Suwon

학번 : 이름 :

```
        "Pusan",
        "Inchon",
        "Suwon",
        "Daegu"
    };

    printf("[도시이름]\n");
    for (int i = 0; i < 5; i++) {
        printf("%s ", city[i]);
    }

    printf("\n[n으로 끝나는 도시만 출력]\n");
    for (int i = 0; i < 5; i++) {
        int len=0;
        while (city[i][len] != '\0') {
            len++;
        }
        if (city[i][len - 1] == 'n') {
            printf("%s ", city[i]);
        }
    }

    printf("\n");

    return 0;
}
```

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

int main(void) {
    char city[5][7] = {
        "Seoul",
        "Pusan",
        "Inchon",
        "Suwon",
        "Daegu"
    };

    char (*p)[7] = city;

    printf("[도시이름]\n");
```

학번 : 이름 :

```
for (int i = 0; i < 5; i++) {
    printf("%s ", *(p + i));
}

printf("\n[n으로 끝나는 도시만 출력]\n");
for (int i = 0; i < 5; i++) {
    char *name = *(p + i);

    if (*(name + strlen(name) - 1) =
= 'n') {
        printf("%s ", name);
    }
}

printf("\n");

return 0;
}
```