Clickseo.com

C Programming

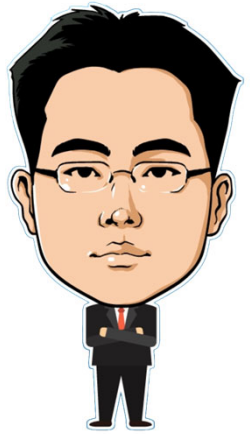
포인터

(Pointers)

Seo, Doo-Ok

Clickseo.com

clickseo@gmail.com



목 차

백문이불여일타(百聞而不如一打)

- 포인터의 이해
- 다양한 포인터

포인터의 이해

백문이불여일타(百聞而不如一打)

- **포인터의 이해**
  - 포인터형 변수
  - 포인터 연산
- **다양한 포인터**

포인터형 변수 (1/9)

- **주소 연산자(&)**
  - 변수와 배열 원소에만 적용한다.
    - 산술식이나 상수 그리고 레지스터 변수에는 주소 연산자를 사용할 수 없다.

```
#include <stdio.h>
int main(void)
{
    int     a;

    &100;               // error C2101: 상수에 '&'이(가) 있습니다.
    &(a + 1);           // error C2102: '&'에 I-value가 있어야 합니다.

    // error C2103: 레지스터 변수에 '&'이(가) 있습니다.
    register  int r;
    &r;

    return 0;
}
```

```
#include <stdio.h>
int main(void)
{
    int     a, b;
    printf("%p %p ₩n", &a, &b);
    return 0;
}
```

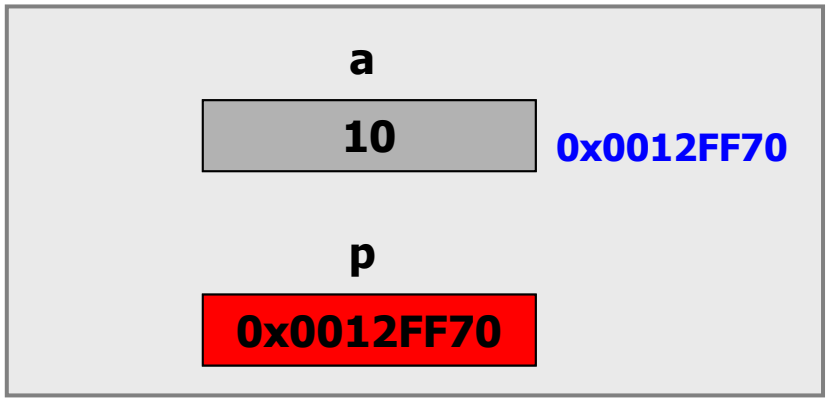
```
Microsoft Visual Studio 디버그
000000060348FF544 000000060348FF564

C:\Users\click\OneDrive\문서\cClickseo\x64\
이 창을 닫으려면 아무 키나 누르세요...
```

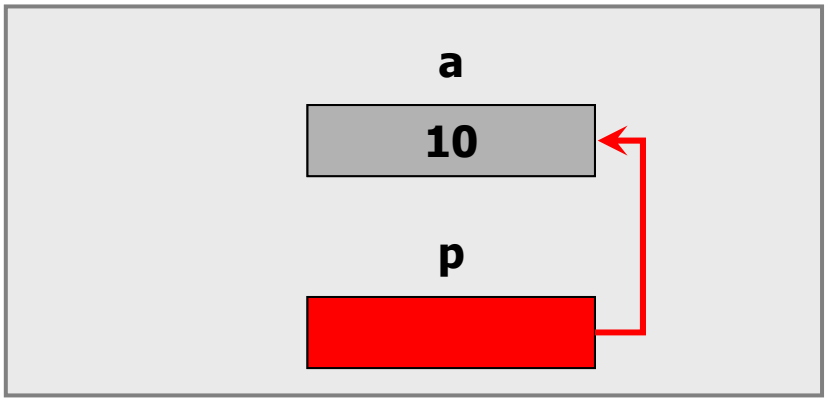
포인터형 변수 (2/9)

- **포인터형 변수:** 물리적/논리적 표현
  - 포인터형 변수: 메모리 공간의 주소를 저장할 수 있는 변수

```
int     a = 10;          // * : 간접 연산자 또는 역 참조 연산자
int*    p = &a;          // *p == a          // p == &a
```

  - 물리적인 표현과 논리적인 표현

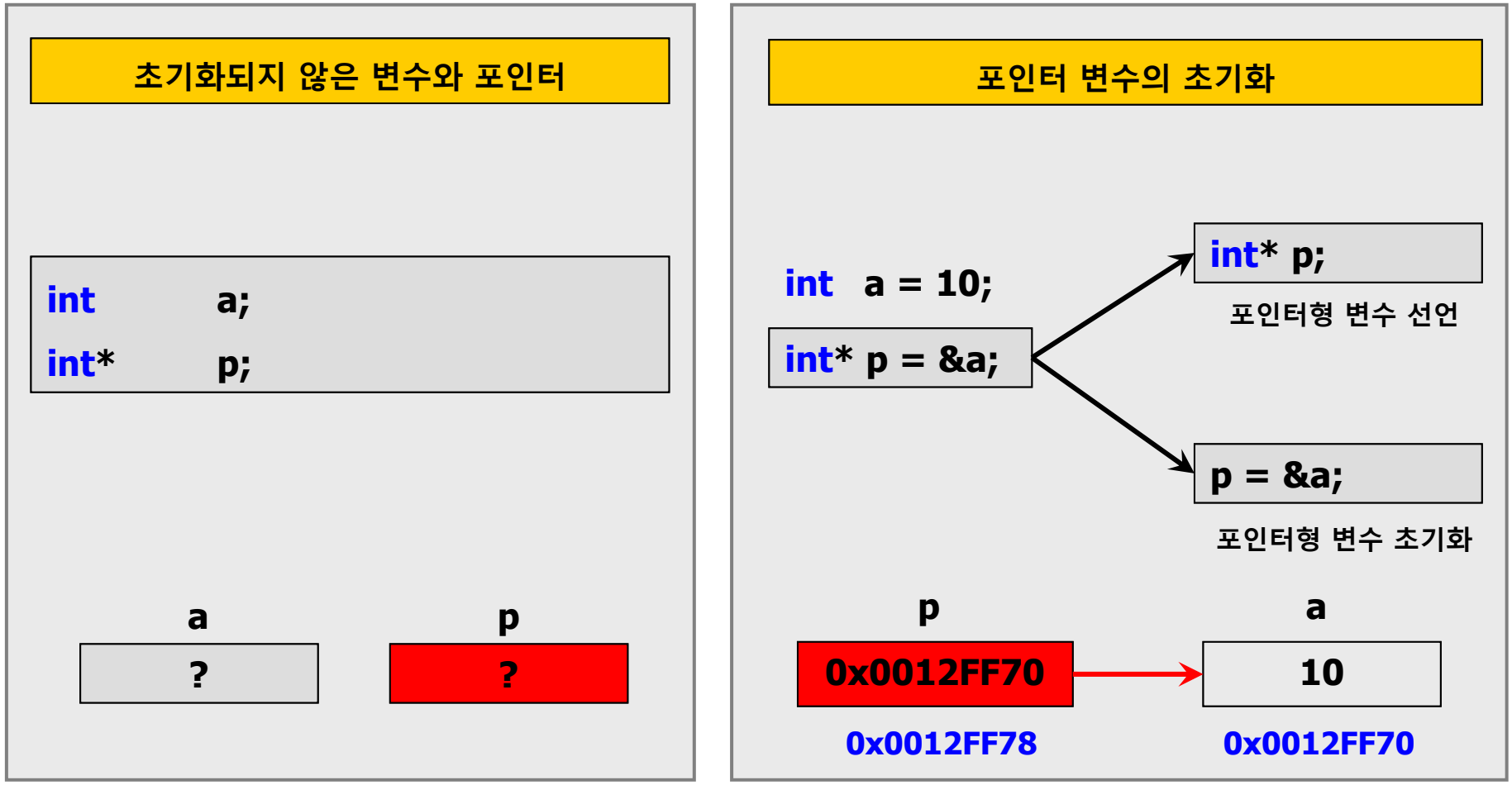




"포인터형 변수는 메모리 주소 이외에
어떠한 값도 저장하지 않는다는 것을 절대 잊으면 안 된다."

포인터형 변수 (3/9)

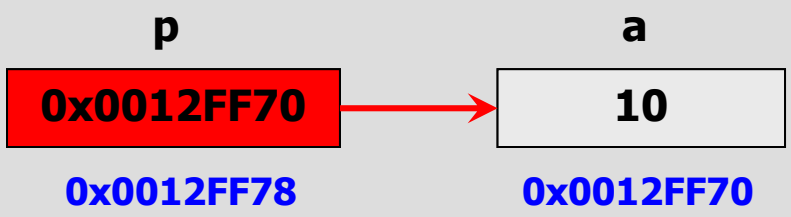
- **포인터형 변수:** 선언 및 초기화



포인터형 변수 (4/9)

- **포인터형 변수:** 간접 접근

```
// 일반 변수와 포인터형 변수 선언
int         a;
int*        p;

// 일반 변수와 포인터형 변수의 초기화
a = 10;
p = &a;
```

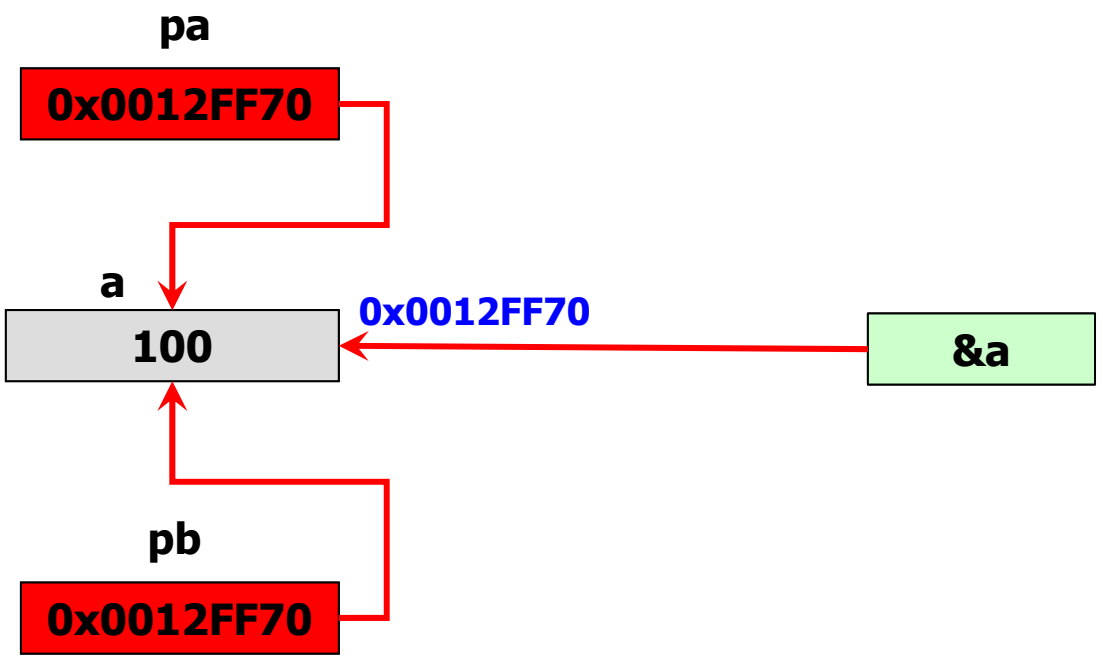


*p	a	10	// 변수 a의 데이터
p	&a	0x0012FF70	// 변수 a의 주소

```
a = a + 1;          // a++;
*p = *p + 1;        // (*p)++;
```

포인터형 변수 (5/9)

- **포인터형 변수:** 다중 포인터
  - 변수에 대한 다중 포인터



포인터형 변수 (6/9)

- **포인터형 변수에 왜 자료형을 지정하는가?**

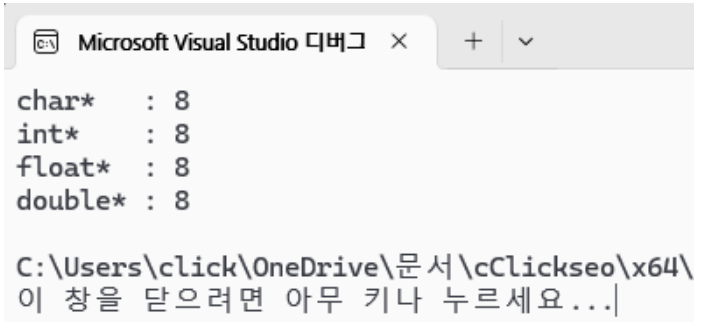
```
// 컴파일러는 아래 문장을 어떻게 처리할까?
// 즉, 메모리에 몇 바이트 씩을 할당할까?

char*       pc;
int*        pi;
float*      pf;
double*     pd;
```

"포인터형 변수의 자료형은 포인터 변수가 가리키는 대상체의 크기(자료형),
즉, 저장하고 있는 메모리 주소를 간접 접근하여
어떤 형태(자료형)로 접근(해석)할 것인가를 의미한다."

포인터형 변수 (7/9)

예제 4-1: 포인터형 변수 Visual Studio Community 2022 (64bits)

```c
#include <stdio.h>
int main(void)
{
        char*   pc;
        int*    pi;
        float*  pf;
        double* pd;

        // %zu -- size_t  (unsigned int)
        // %zd -- ssize_t (signed int)
        printf("char*   : %zd \n", sizeof(char*) );
        printf("int*    : %zd \n", sizeof(int*) );
        printf("float*  : %zd \n", sizeof(float*) );
        printf("double* : %zd \n", sizeof(double*) );

        return 0;
}
```



포인터형 변수 (8/9)

예제 4-1: 포인터형 변수 GNU/Linux GCC (64bits)

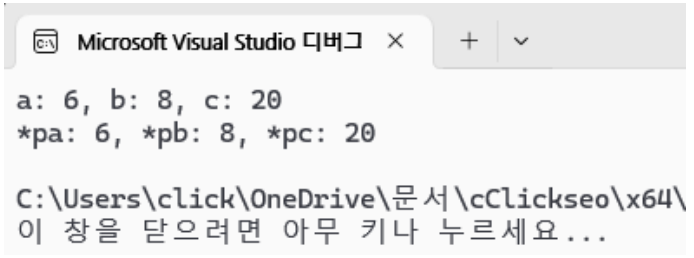
```c
#include <stdio.h>
int main(void)
{
        char*   pc;
        int*    pi;
        float*  pf;
        double* pd;

        // warning: format '%d' expects argument of type 'int',
        // but argument 2 has type 'long unsigned int'
        // printf("char * : %d \n", sizeof(char *) );

        printf("char*   : %ld \n", sizeof(char*) );
        printf("int*    : %ld \n", sizeof(int*) );
        printf("float*  : %ld \n", sizeof(float*) );
        printf("double* : %ld \n", sizeof(double*) );

        return 0;
}
```

```
clickseo@clickseo-VirtualBox: ~/gcc
clickseo@clickseo-VirtualBox:~/gcc$ gcc pointer.c
clickseo@clickseo-VirtualBox:~/gcc$ ./a.out
char *   : 8
int *    : 8
float *  : 8
double * : 8
clickseo@clickseo-VirtualBox:~/gcc$
```

포인터형 변수 (9/9)

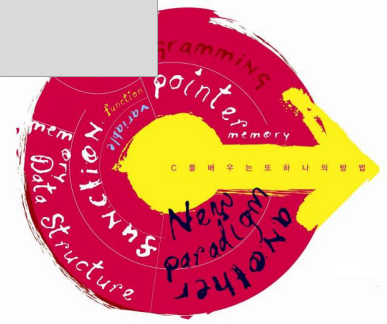
예제 4-2: 일반 변수와 포인터형 변수 -- 직접 접근과 간접 접근

```c
#include <stdio.h>
int main(void)
{
        int     a, b, c;
        int     *pa, *pb, *pc;

        a = 6;
        b = 2;

        pa = &b;
        pb = pa;
        pc = &c;

        pa = &a;
        *pb = 8;

        *pc = *pa;
        *pc = a + *pb + *&c;

        printf("a: %d, b: %d, c: %d\n",  a, b, c );
        printf("*pa: %d, *pb: %d, *pc: %d\n", *pa, *pb, *pc );

        return 0;
}
```



포인터의 이해

포인터 연산



포인터 연산 (1/2)

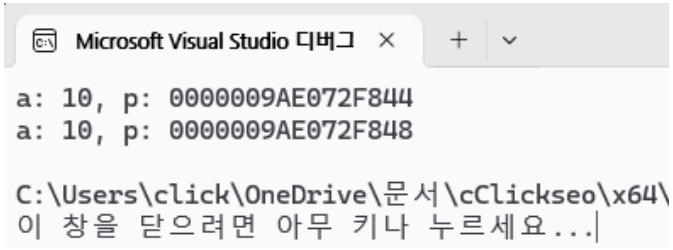
- **포인터 연산**
  - 연산의 대상: 메모리 주소
    - 정수 연산만 가능
    - 사용 가능 연산자: + , - , ++ , -- , > , >= , < , <= , == , !=
  - 포인터가 가리키는 대상체(자료형)의 크기 만큼 연산이 일어난다.

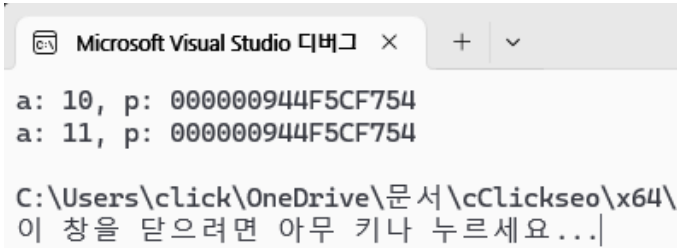
```
int         a = 10;
int*        p = &a;


p++;        //  실제로 증가되는 대상은?
```

포인터 연산 (2/2)

예제 4-3: 포인터 연산 -- 증감 연산자

```
#include <stdio.h>

#if 1
int main(void)
{
        int     a = 10;
        int*    p = &a;

        printf("a: %d, p: %p \n", a, p);
        p++;            // p = p + 1;
        printf("a: %d, p: %p \n", a, p);
        return 0;
}

#elif 0
int main(void)
{
        int     a = 10;
        int*    p = &a;

        printf("a: %d, p: %p \n", a, p);
        (*p)++;         // a++;
        printf("a: %d, p: %p \n", a, p);
        return 0;
}
#endif
```

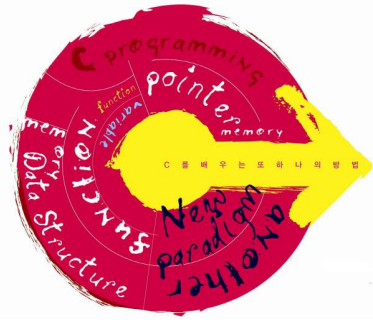




다양한 포인터

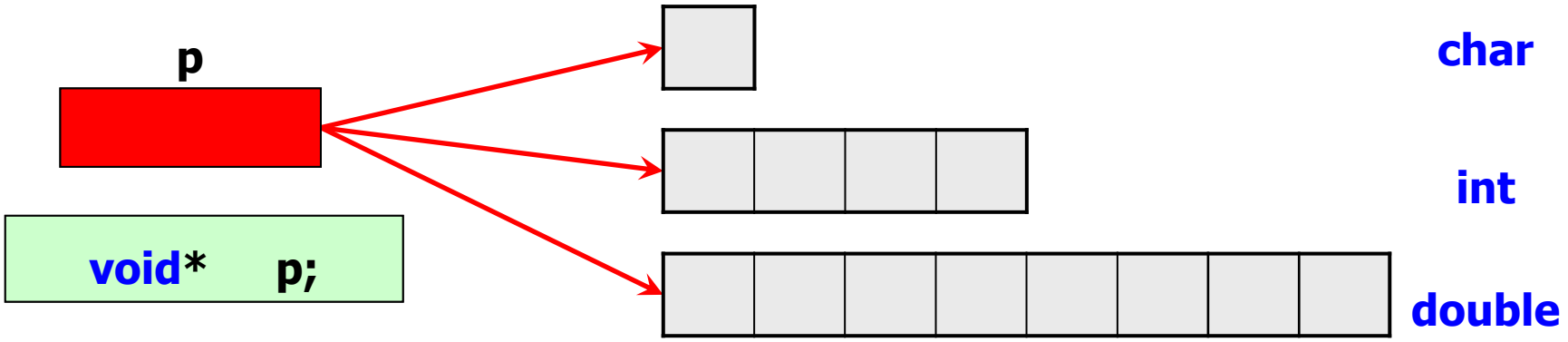
백문이불여일타(百聞而不如一打)

- 포인터의 이해
- **다양한 포인터**
  - void형 포인터
  - NULL 포인터
  - 다중 포인터



void형 포인터 (1/4)

- **void형 포인터**
  - 어떤 대상체(자료형)의 메모리 주소 든지 저장할 수 있는 포인터
    - 포인터형 변수 선언 시 간접 접근 방법을 지정하지 않은 포인터
      - 간접 접근으로 대상체의 메모리를 접근 할 때는 반드시 형 변환이 이루어져야 한다.



void형 포인터 (2/4)

- **void형 포인터:** 다양한 자료형
  - 다양한 자료형의 메모리 공간
    - 어떠한 형 변환 없이도 다양한 자료형의 메모리 공간의 주소 값을 저장할 수 있다.

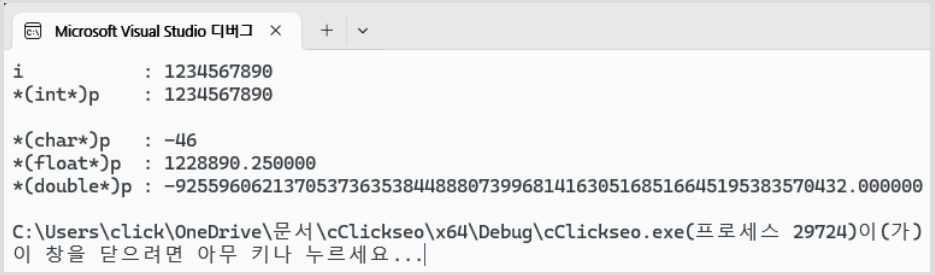
```c
#include <stdio.h>
int main(void)
{
    char        c;
    int         i;
    float       f;
    double      d;

    // warning C4133: '=': 'char *'과(와) 'int *' 사이의 형식이 호환되지 않습니다.
    // int* p = &c;     // int* p = (int*)&c;

    void      *p;

    p = &c;
    p = &i;
    p = &f;
    p = &d;

    return 0;
}
```

```c
#include <stdio.h>
int main(void)
{
    char*  pc;
    int*   pi;

    // warning C4133:
    // '=': 'char *'과(와) 'int *' 사이의 형식이 호환되지 않습니다.
    int*   p;
    p = pc;             // p = (int*)pc;
    p = pi;

    return 0;
}
```

void형 포인터 (3/4)

- **void형 포인터: 간접 접근과 형 변환**
  - 다양한 자료형의 간접 접근
    - void 포인터를 이용하여 간접 접근시에는 반드시 형 변환이 이루어져야 한다.
      - 즉, 대상체의 직접 접근 방법(자료형)과 일치해야 의미 있는 값을 가져 올 수 있다.

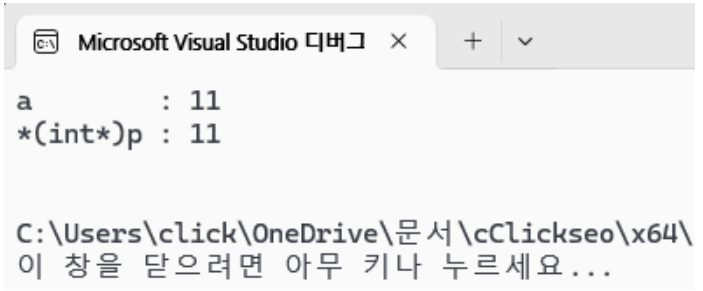
```c
#include <stdio.h>
int main(void)
{
    int    i = 1234567890;

    void*  p = &i;
    printf("i            : %d \n", i );
    printf("*(int*)p     : %d \n\n", *(int*)p );

    printf("*(char*)p    : %d \n", *(char*)p );
    printf("*(float*)p   : %f \n", *(float*)p );
    printf("*(double*)p  : %f \n", *(double*)p );
    return 0;
}
```



void형 포인터 (4/4)

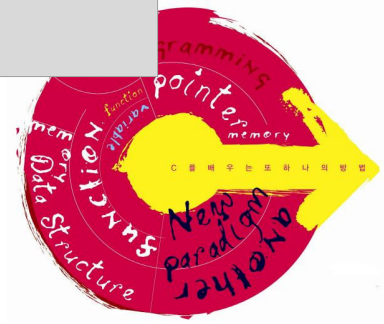
예제 4-4: void형 포인터 -- 증감 연산자

```
#include <stdio.h>
int main(void)
{
        int     a = 10;
        void*   p = &a;

        // error C2100: 간접 참조가 잘못되었습니다.
        // error C2036: 'void *' : 알 수 없는 크기입니다.
        // (*p)++;      // *p = *p + 1;

        // *(int*)p = *(int*)p + 1;
        (*(int*)p)++;

        printf("a         : %d \n", a);
        printf("*(int*)p  : %d \n\n", *(int*)p);

        return 0;
}
```



다양한 포인터

NULL 포인터, 다중 포인터

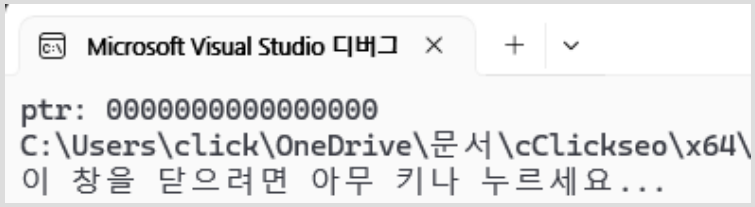


NULL 포인터

- **NULL 포인터**
  - 매크로 상수: 널 포인터 상수

```
#define NULL (void*) 0
```

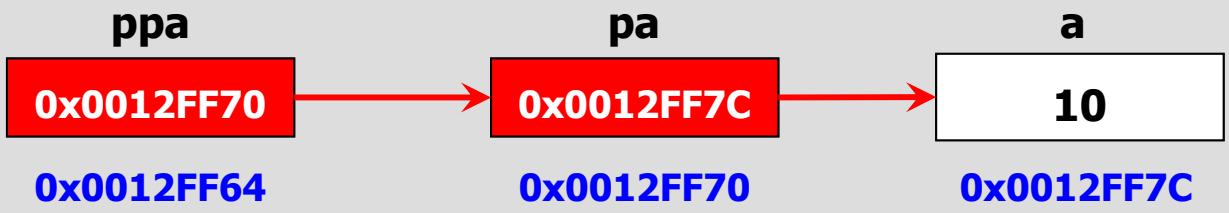
```
#include <stdio.h>          // NULL
int main(void)
{
    // NULL : NULL 포인터
    int*    ptr = NULL;

    printf("ptr: %p", ptr);

    return 0;
}
```



다중 포인터

- **이중 포인터(Double Pointer)**
  - 포인터에 대한 포인터

```
#include <stdio.h>
int main(void)
{
    int    a = 10;
    int*   pa;
    int**  ppa;

    pa = &a;
    ppa = &pa;

    printf("a    : %d, &a   : %p \n", a, &a );
    printf("*pa  : %d, pa   : %p \n", *pa, pa );
    printf("**ppa: %d, *ppa: %p \n", **ppa, *ppa );

    return 0;
}
```




```
Microsoft Visual Studio 디버그
a    : 10, &a   : 000000812534FA04
*pa  : 10, pa   : 000000812534FA04
**ppa: 10, *ppa: 000000812534FA04

C:\Users\click\OneDrive\문서\cClickseo\x64\
이 창을 닫으려면 아무 키나 누르세요...
```

참고문헌

[1] 서현우, “혼자 공부하는 C 언어: 1:1 과외 하듯 배우는 프로그래밍 자습서”, 한빛미디어, 2023.

[2] Paul Deitel, Harvey Deitel, "C How to Program", Global Edition, 8/E, Pearson, 2016.

[3] Kamran Amini, 박지윤 번역, “전문가를 위한 C : 동시성, OOP부터 최신 C, 고급 기능까지!”, 한빛미디어, 2022.

[4] 서두옥, “(열혈강의) 또 하나의 C : 프로그래밍은 셀프입니다”, 프리렉, 2012.

[5] Behrouz A. Forouzan, Richard F. Gilberg, 김진 외 7인 공역, “구조적 프로그래밍 기법을 위한 C”, 도서출판 인터비젼, 2004.

[6] Brian W. Kernighan, Dennis M. Ritchie, 김석환 외 2인 공역, “The C Programming Language”, 2/E, 대영사, 2004.

[7] "C reference", cppreference.com, 2024 of viewing the site, https://en.cppreference.com/w/c.