12장 포인터의 이해

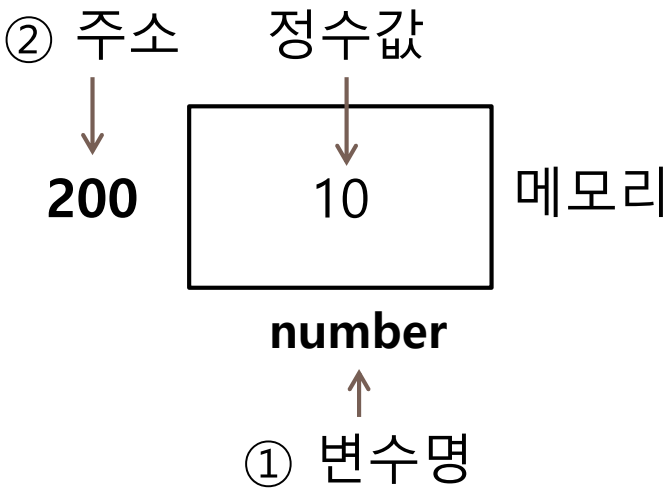
차례

1. 포인터란 무엇인가?
2. 포인터와 관련 있는 연산자: &연산자, *연산자

포인터의 응도

- 변수를 접근하는 방법
  - 변수명으로 접근(직접 참조) -> 지금까지 사용한 방법
  - 주소로 접근(간접 참조) -> 포인터를 이용하는 방법

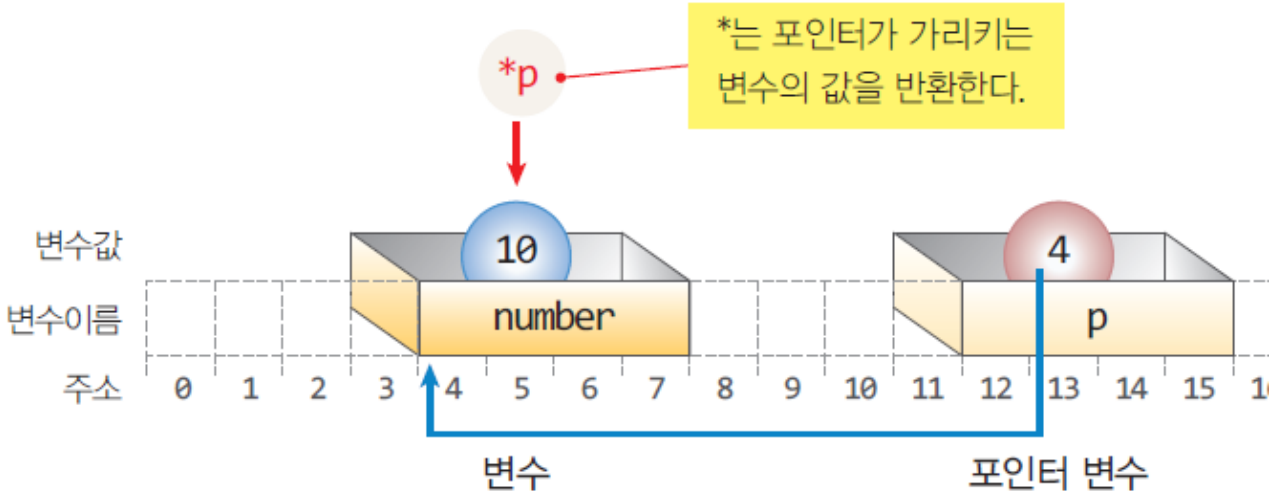
```
int number = 10;
```



간접 참조 연산자 *

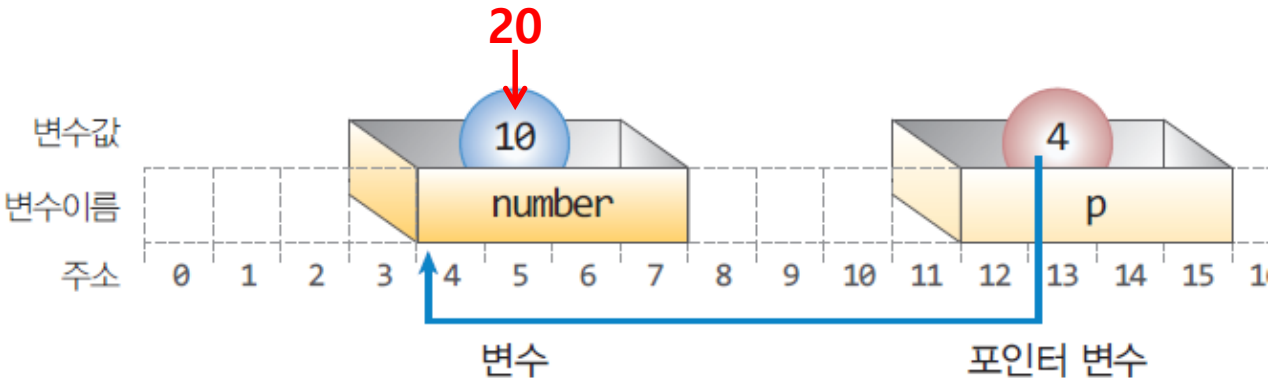
- 간접 참조연산자 * : 포인터(주소)가 가리키는 변수에 저장된 값을 구해주는 연산자
- ***p** : 포인터(주소) p에 저장된 주소가 가리키는 메모리 공간의 값

```
int number = 10;
int* p = &number;
printf("%d₩n", number);          // 10이 출력됨 -> 직접참조
printf("%d₩n", *p);              // 10이 출력됨 -> 간접참조
```



간접 참조 연산자 *


```
int number = 10;
int* p = &number;
printf("%d₩n", *p);          // 10이 출력됨 -> 간접참조
*p = 20;                     // p가 가리키는 공간에 20을 대입
printf("%d₩n", *p);          // 20이 출력됨 -> 간접참조
```



간접 참조 연산자 *

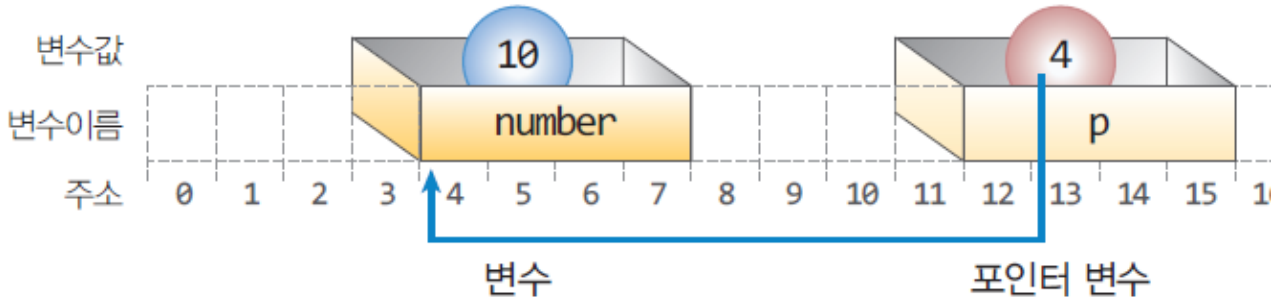
- 간접 참조연산자 * 뒤에는 포인터와 주소 모두 올 수 있음

```
int number = 10;
printf("%d₩n", *&number);        // *(&number) -> *(주소) -> 10
int* p = &number;
printf("%d₩n", *p);              // *포인터 -> 10
```

간접 참조 연산자 *

- *연산자는 **포인터에 저장된 주소로부터 포인터가 가리키는 자료형의 크기만큼** 읽거나 쓰기를 수행함

```
int number = 10;
int* p = &number;
printf("%d₩n", *p);                    // 10 -> 4번지부터 4바이트만큼 읽음
*p = 20;
printf("%d₩n", *p);                    // 20 -> 4번지부터 4바이트만큼 읽음
```

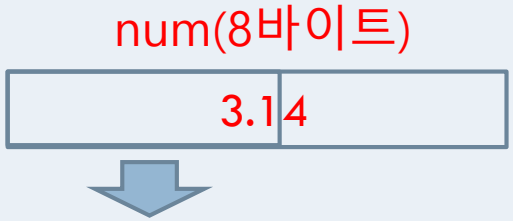


간접 참조 연산자 *

- 잘못된 사용 예

```
#include<stdio.h>
int main(void)
{
    double num = 3.14;
    int* pnum=&num;
    printf("num:%d₩n", *pnum);
    return 0;
}
```

num:1374389535



*pnum ->저장된 주소부터 4바이트

다양한 포인터의 형이 존재하는 이유

- 주소 값이 정수임에도 불구하고 int형 변수에 저장하지 않는 이유는 int형 변수에 저장하면 메모리 공간의 접근을 위한 간접 참조연산(*)이 불가능하기 때문
- 다양한 포인터 형을 정의한 이유는 * 연산을 통한 메모리의 접근기준을 마련하기 위함
  - int형 포인터 변수로 * 연산을 통해 메모리(변수) 접근 시 4바이트 메모리 공간에 부호 있는 정수의 형태로 데이터를 읽고 씀
  - double형 포인터 변수로 * 연산을 통해 메모리(변수) 접근 시 8바이트 메모리 공간에 부호 있는 실수의 형태로 데이터를 읽고 씀

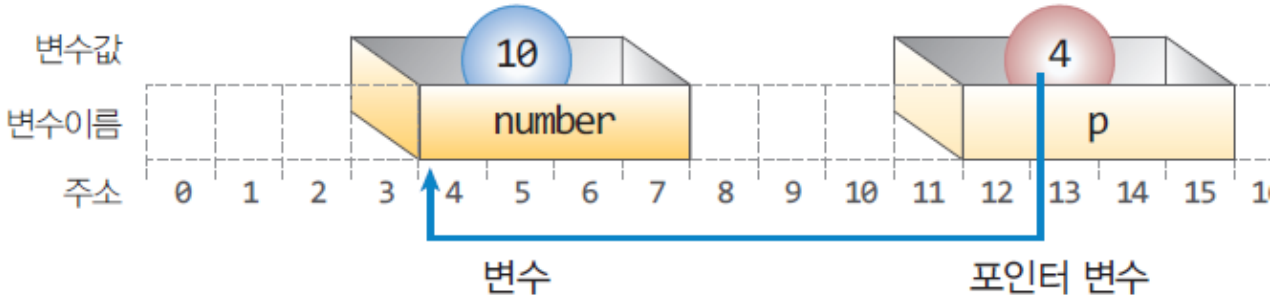
간접 참조 연산자

- 변수를 접근하는 방법 : 변수명으로 접근(직접참조), 포인터로 접근(간접참조)

```
int number = 10;

printf("%d₩n", &number); //4 출력
printf("%d₩n", number);   //10 출력
number = 20;
printf("%d₩n", number);   //20 출력
```

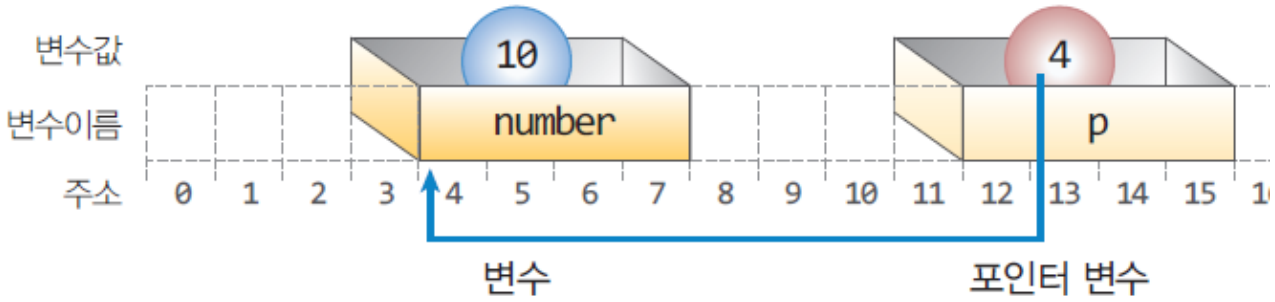
```
int number = 10;
int* p = &number;
printf("%d₩n", p);          //4출력
printf("%d₩n", *p);         //10출력
*p = 20;
printf("%d₩n", *p);         //20출력
```



간접 참조 연산자

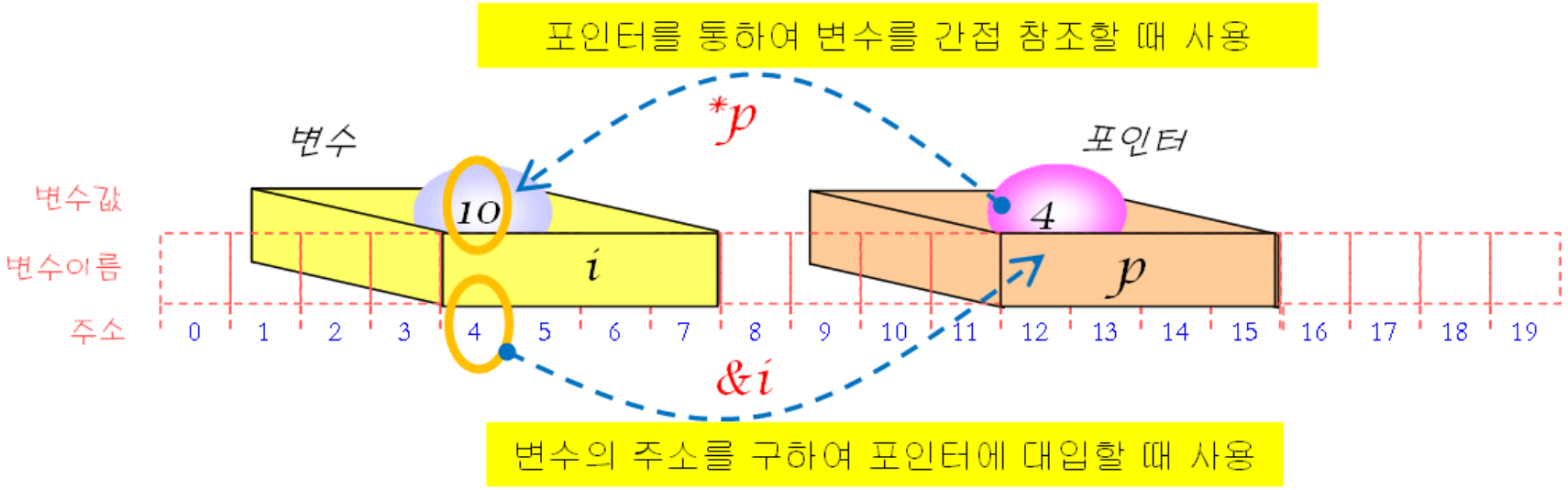
- 포인터도 변수이므로 주소연산자를 사용가능

```
int number = 10;
int* p = &number;
printf("%d₩n", p);          //4 출력
printf("%d₩n", *p);         //10 출력
printf("%d₩n", &p);         //12 출력
```



& 연산자와 , * 연산자

- & 연산자: 변수의 주소를 반환
- * 연산자: 포인터가 가리키는 곳의 내용물을 반환



예제 1


```
#include <stdio.h>
int main(void)
{
    int number = 10;
    int* p;
    p = &number;                                    // 변수와 포인터 연결

    printf("&number = %u₩n", &number);    // 변수의 주소 출력
    printf("p = %u₩n", p);                          // 변수의 주소 출력
    printf("number = %d₩n", number);        // 변수명을 통한 값 출력
    printf("*p = %d₩n", *p);                        // 포인터를 통한 값 출력
    printf("*&number = %d₩n", *&number); // (*(&number))
    return 0;
}
```

```
&number = 1245024
p = 1245024
number = 10
*p = 10
*&number = 10
```

예제2


```c
#include <stdio.h>
int main(void)
{
    int number = 10;
    int* p;
    p = &number;
    printf("number = %d₩n", number);
    printf("number = %d₩n", *p);
    *p = 20;                                    //number = 20;
    printf("number = %d₩n", number);
    printf("number = %d₩n", *p);
    return 0;
}
```

```
number = 10
number = 10
number = 20
number = 20
```

예제3

num1:130,num2:70

```
#include<stdio.h>
int main(void)
{
	int num1 = 100, num2=100;
	int* pnum;

	pnum = &num1;
	(*pnum) += 30;			// (*pnum) = (*pnum) + 30;

	pnum = &num2;			//pnum은 변수이므로 값변경가능
	(*pnum) -= 30;

	printf("num1:%d,num2:%d₩n", num1, num2);

	return 0;
}
```

포인터 사용시 주의점 1

- 초기화가 안된 포인터를 사용하면 안됨 -> 실행 중에 오류 발생

```
int main(void)
{
    int* p;          // 포인터 p는 초기화가 안되어 있음(쓰레기값)
    *p = 100;        // 위험한 코드
    return 0;
}
```

포인터 사용시 주의점 2

- 포인터의 초기값이 없는 경우에는 기호상수 NULL로 초기화
- 기호상수 NULL은 stdio.h안에 0으로 정의되어 있음
- NULL 포인터를 가지고 간접 참조하면 컴퓨터가 자동으로 오류를 감지함
- 포인터의 유효성 여부 판단이 쉽다.

```
int* p = NULL;
```

포인터 사용시 주의점 3

- 포인터의 타입과 변수의 타입은 일치해야 함

```
#include <stdio.h>
int main(void)
{
    int i;
    double* pd;
    pd = &i;      //오류! double형 포인터에 int형 변수의 주소를 대입
    *pd = 36.5;
    return 0;
}
```

포인터 사용시 주의점 4

- * 는 사용되는 곳에 따라 곱셈기호, 간접 참조연산자, 포인터기호로 사용됨 -> 문장에 맞게 해석해야 함

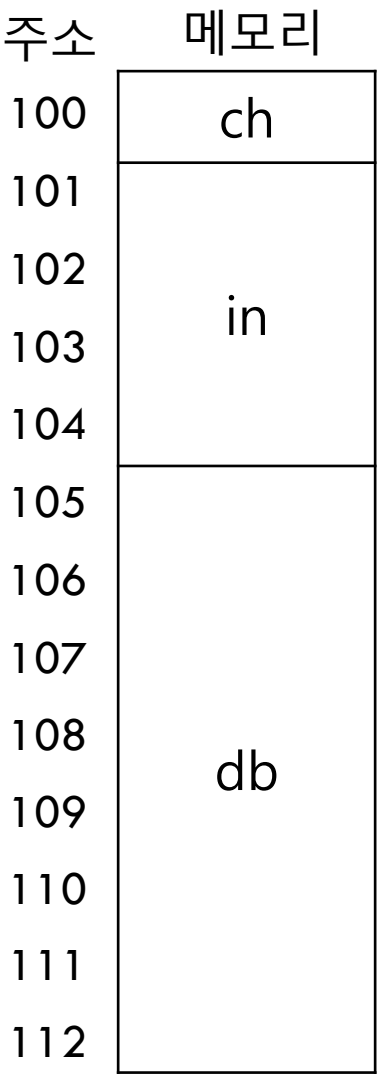
```
#include <stdio.h>
int main(void)
{
    int i;
    double* pd;          //*는 포인터변수 선언
    pd = &i;
    *pd = 36.5;          //*는 간접참조연산자
    i = i*10;            //*는 곱셈기호
    return 0;
}
```

실습과제 1

- 아래의 변수가 우측 그림처럼 메모리가 할당될 때 다음 표의 빈칸을 채우시오.
- 힌트 : *&ch -> (*(&ch))

```
char ch = 'A';
int in = 10;
double db = 3.4;
```

수식	결과값	결과값의 자료형
&ch		
&in		
&db		
*&ch		
*&in		
*&db		



실습과제2

- 아래 코드에서 변수이름을 사용하지 말고 포인터를 사용하여 같은 결과가 나오도록 코드를 수정하시오(예제1번 참조)

```
#include <stdio.h>
int main(void)
{
    int a = -100;
    char b = 'A';
    double c = 3.14;
    printf("int형 변수 a의 값은 : %d₩n", a);
    printf("char형 변수 b의 값은 : %c₩n", b)
    printf("double형 변수 c의 값은 : %lf₩n", c);
    return 0;
}
```

int 형 변수 a의 값은 : -100
char형 변수 b의 값은 : A
double형 변수 c의 값은 : 3.14

실습과제3

- 아래 코드에 문제점을 자세히 설명하라.

실행중단됨

```
#include<stdio.h>
int main(void)
{
	int* ptr = (int*)125;
	*ptr = 10;
	printf("%d₩n", *ptr);
	return 0;
}
```

실습과제 4

- 아래코드를 포인터를 이용한 간접 참조 방식의 코드로 필요한 코드를 추가 또는 수정하시오. 실행 결과는 같아야 한다.

두정수의 합: 300

```
#include <stdio.h>
int main(void)
{
    int a = 100, b=200;
    int sum;

    sum = a + b;  //포인터를 이용하여 수정
    printf("두정수의 합 : %d₩n", sum); //포인터를 이용하여 수정
    return 0;
}
```

실습과제 5

- 교재 284페이지 문제1번과 유사하게 자유롭게 변형하여 새로운 문제를 만들고 푸시오.
- 똑같은 문제를 풀면 0점 처리함

실습과제 6

- 교재 284페이지 문제2번과 유사하게 자유롭게 변형하여 새로운 문제를 만들고 푸시오.
- 똑같은 문제를 풀면 0점 처리함

12장 정리문제

1) 메모리 주소가 무엇인가?
2) 변수의 메모리 주소를 구하는 방법은?
3) 포인터의 의미를 설명하라
4) 포인터변수가 왜 필요한가?
5) 포인터변수가 메모리에 할당될 때 메모리 크기는 몇 바이트인가?
6) 포인터 선언은 자료형 *변수명; 처럼하는데 여기서 자료형의 의미를 설명하라
7) 정수 100과 주소 100을 어떻게 구분하는가?
8) 포인터를 이용하여 메모리 공간에 저장된 값을 접근하는 방법을 설명하라.
9) swap함수 예제가 의미하는 바를 설명하시오.

과제제출방법

- 소스코드, 라인단위의 주석, 실행결과를 포함하는 pdf파일을 작성한 후 eclass 과제 게시판에 업로드, **반드시 하나의 pdf파일로 업로드할 것**
- 기한 : 과제 게시판에 마감시간 참조
- 실행결과를 캡쳐할 때 글자를 알아보기 쉽게 확대해서 캡쳐할 것.
- 소스코드의 첫 부분은 아래처럼 제목,날짜,작성자(학번,이름)를 작성할 것

```
// ********************************************
//    제   목 : 정수 4개의 평균을 구하는 프로그램
//    날   짜 : 2023년 9월10일
//    작성자 : 15010101 홍길동
// ********************************************

// 소스코드 작성
```