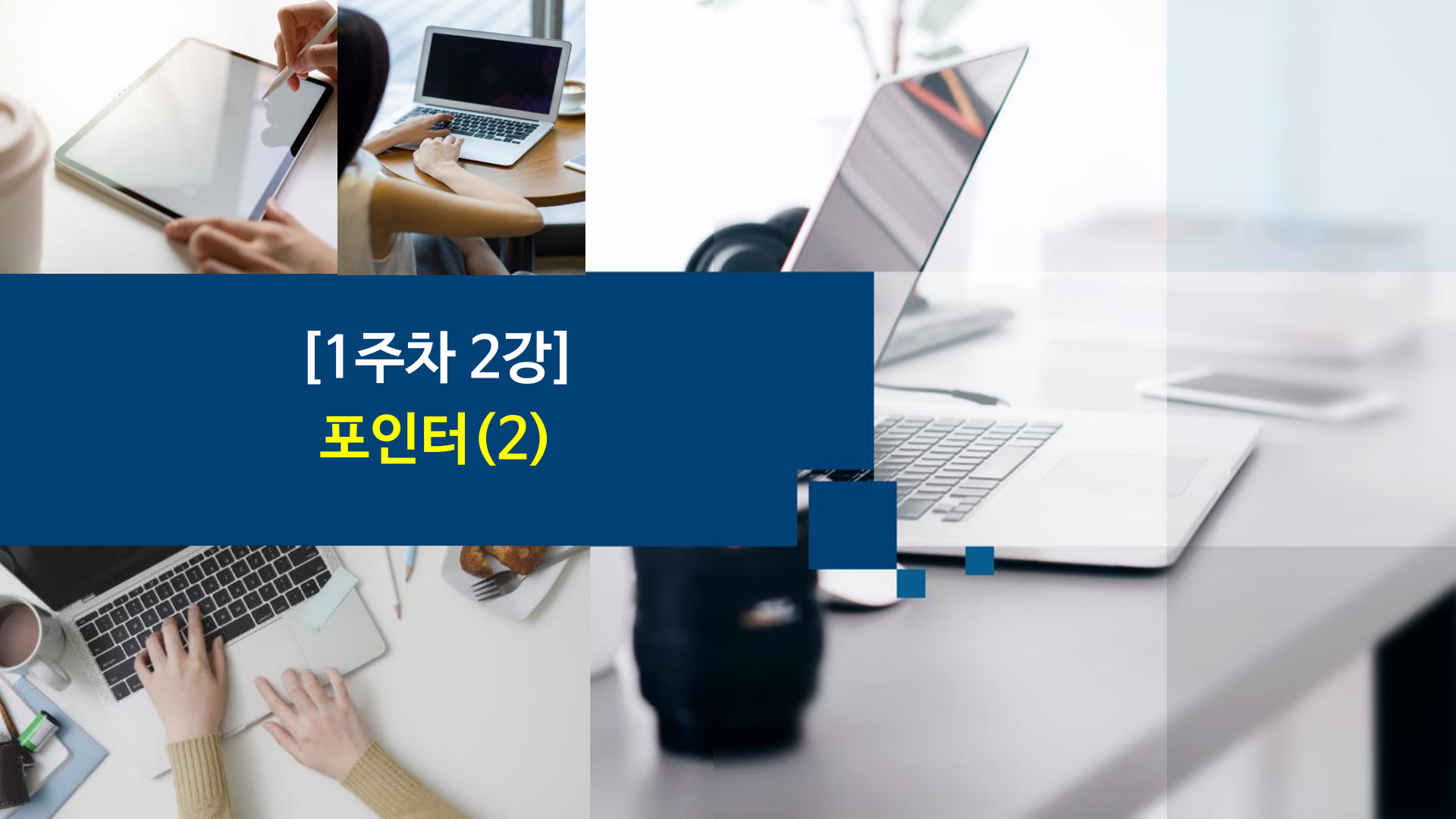


[1주차 2강]
포인터(2)

학습 내용

- 9.2 포인터 선언과 사용

C 프로그래밍 및 실습

1. 포인터 개요
2. **포인터 선언과 사용**
3. 배열과 포인터
4. 포인터 연산
5. 포인터 인자와 주소 반환
6. 포인터 배열
7. 다중 포인터

2. 포인터 선언과 사용

포인터 (변수) 선언

- 구문 : 변수 명 앞에 * **(참조연산자)**만 덧붙이면 됨
  - ✓ 기존의 자료형 표시 + 포인터라는 표시
  - ✓ 예)

```
char    *pch;
int     *pnum;
```

  - ✓ pch와 pnum은 똑같이 주소를 저장하지만

    대상의 자료형이 다르기 때문에 **다른 자료형**으로 취급
  - ✓ pch는 문자형 포인터 (변수)이고 pnum은 정수형 포인터 (변수)

2. 포인터 선언과 사용

초기화

- 일반 변수 초기화 형태와 동일

```
int num, *pnum = &num;
```

✓ (주의!!) num이 먼저 선언 되어야 함

```
int *pnum = &num, num;          // 컴파일 오류
```

포인터 변수 선언의 다양한 형태

✓ 동일 기본 자료형(int)에서 파생된 자료형의 변수는 모아서 선언 가능

```
int *pnum1, num1=10, *pnum2, num2, arr[10];
```

✓ 그러나, 가독성 때문에 추천 안 함

2. 포인터 선언과 사용

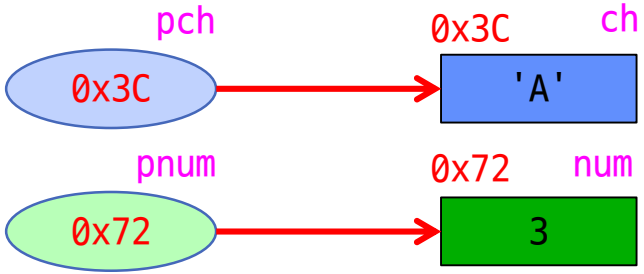
포인터 대입 (연결)

- 포인터 (변수)에 **주소를 대입**하여 특정 변수와 연결시키는 것을 **"가리킨다"**라고 표현하고, 그림에서는 **화살표 →** 로 표시

```
char ch = 'A', *pch;
int num = 3, *pnum;

pch = &ch;⇨ pch에 변수 ch의 주소 대입(연결)
pnum = &num;          ⇨ pnum에 변수 num의 주소 대입(연결)

printf("%p %c\n", pch, ch); ⇨ %p: 주소 출력 서식
printf("%p %d\n", pnum, num);
```



실행 예시

```
001EA03C A
001EA072 3
```

2. 포인터 선언과 사용

포인터 참조

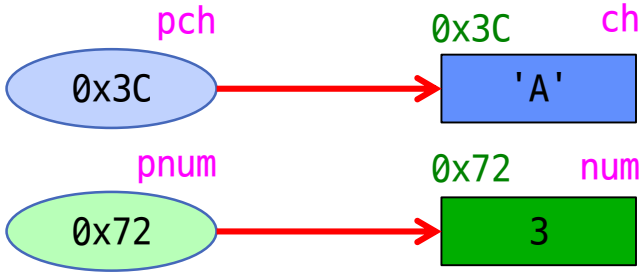
- 포인터 (변수)가 가리키는 변수에 접근하는 것
- 참조 연산자 * (간접연산자, 포인터연산자라고도 부름)를 사용

예) *pch : 포인터 pch가 가리키는 변수, 0x3C번지에 저장된 값

```
char ch = 'A';
char *pch = &ch;

int num = 3, *pnum = &num;

printf("%p %c\n", pch, *pch);
printf("%p %d\n", pnum, *pnum);
```



실행 예시

```
001EA03C A
001EA072 3
```

2. 포인터 선언과 사용

참조 연산자를 이용한 대입 예시

- *pch = 'B' 의 의미 : pch가 가리키는 공간에 'B' 대입
- *pch = 'B'는 ch = 'B'와 동일한 기능 수행

  전자는 **간접 접근**, 후자는 **직접 접근**

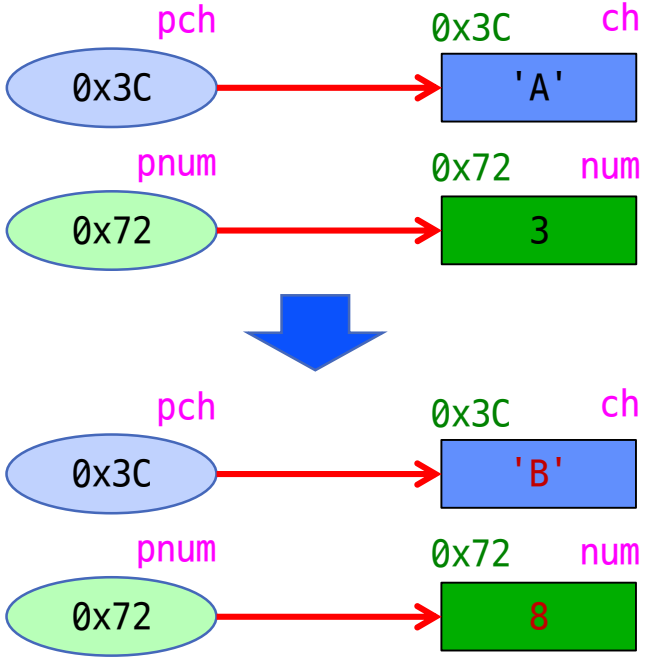
```
char ch = 'A', *pch = &ch;
int num = 3, *pnum = &num;

*pch = 'B';
*pnum += 5;

printf("%c %d\n", ch, num);
```

실행 결과

```
B 8
```



2. 포인터 선언과 사용

참조 연산자 추가 예시

- ✓ *pnum은 정수를 나타내므로, 정수를 사용하는 어떤 형태든 가능
- ✓ 단, 참조연산자와 다른 연산자와의 우선 순위에 주의해서 사용

```
int num = 3, *pnum = &num;

*pnum = *pnum / 2 + 4; ⇨ 정수 연산: num에 num/2+4 = 5 대입

if( *pnum == 5)    ⇨ 정수 비교: ( num == 5 )
  ++*pnum;                  ⇨ 정수 연산: ++(*pnum) ⇨ ++num
printf("%d", *pnum);        ⇨ 함수의 인자로 사용
```

실행 결과

```
6
```

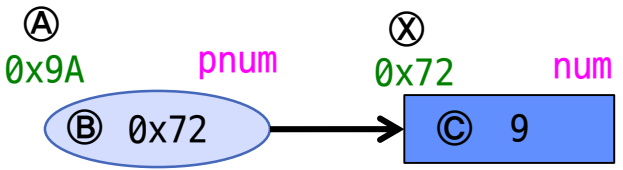
2. 포인터 선언과 사용

포인터와 관련한 두 연산자 정리 (pnum 기준)

- 주소연산자(&) : 해당 변수의 주소 값 (그림의 Ⓐ)
- 변수 이름 : 변수 영역 또는 변수에 저장된 값 (그림의 Ⓑ)
- 참조연산자(*) : 포인터가 가리키는 변수(그림의 Ⓒ)
- **(주의)** pnum과 &num의 값은 동일하지만, **지칭하는 부분은 전혀 다름**

```
int num = 9, *pnum = &num;

printf("%p\n", &pnum);   ⇨ Ⓐ 0x9A
printf("%p\n", pnum);          ⇨ Ⓑ 0x72
printf("%d\n", *pnum);   ⇨ Ⓒ 9
printf("%p\n", &num);          ⇨ Ⓧ 0x72
```

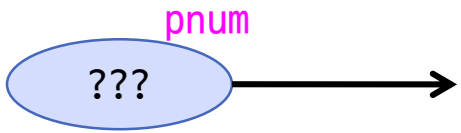


포인터를 이해하고 학습하기 위한 가장 좋은 방법은
메모리 그림을 그리는 것이다!!

포인터 주의사항 1 (초기화)

- 선언 후 연결 없이 바로 사용하면?

```
int *pnum ;    // pnum에는 쓰레기 값
*pnum = 9 ;    // 런타임(실행) 오류 발생
```



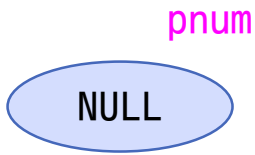

```
int *pnum, num;
pnum = &num; // 반드시 어떤 변수에 연결 후 사용
*pnum = 9;
```

2. 포인터 선언과 사용

널(NULL) 포인터

- **주소 값 0**을 나타내는 특별한 기호
- 아무것도 가리키지 않음을 의미
- NULL의 값은 0이므로, 조건문에서 사용하면 거짓에 해당
- 예기치 못한 오류 방지를 위해 포인터 변수를 NULL로 초기화

```
int *pnum = NULL;
```



포인터 주의사항 2

- & (주소연산자)는 포인터를 포함한 모든 변수에 사용가능

- * (참조연산자)는 포인터 변수에서만 가능
  - ✓ *num (num이 가리키는 변수) 은 정의 되지 않음

```
int num=9, *pnum = &num;
printf("%p %p %d\n", &pnum, pnum, *pnum);
printf("%p %d %d\n", &num, num, *num); // 컴파일 오류
```

2. 포인터 선언과 사용

포인터 주의사항 3 (대입)

- **포인터의 자료형과 연결된 변수의 자료형**은 **일치**해야 한다.
- 서로 다른 자료형의 포인터 간 대입
  - ✓ 문법적으로는 허용이 되기도 하지만 (컴파일 경고만 발생)
  - ✓ **프로그램 오류의 원인**이 됨

```
int num;
char *pch = &num;          // 자료형 불일치

*pch = 4;

printf("%d %d\n", num, *pch);
```

실행 예시
(결과는 다를 수 있음)

```
-858993660 4
```

2. 포인터 선언과 사용

포인터의 크기

- **포인터의 종류(자료형)에 관계없이** 주소를 저장하기 위해 필요한 공간은 동일
  - ✓ 단, 포인터의 크기는 시스템에 따라 다를 수는 있음
- sizeof 연산자를 이용하여 확인해보자.

```
char *pch;
int *pnum;
double *pdnum;

printf("%d\n", sizeof(pch));
printf("%d\n", sizeof(pnum));
printf("%d\n", sizeof(pdnum));
```

실행 결과

```
4
4
4
```

학습 정리

- 포인터 변수는 참조 연산자(*)를 사용하여 **선언**함
- 포인터 변수는 다른 변수에 **연결**을 시킨 후 사용해야 함
- 포인터 변수가 가리키는 변수에 접근하기 위해서는 **참조 연산자(*)**를 사용함
- **NULL**은 주소 값 0을 나타내는 특별 기호로, 아무것도 가리키지 않는다는 것을 의미함