Pointer

Section 1 메모리와 주소
Section 2 일반 변수와 포인터 변수
Section 3 배열과 포인터

일반 변수의 사용 방법과 개념

- **일반 변수의 메모리 할당 (선언)**

주소를 저장하는 변수

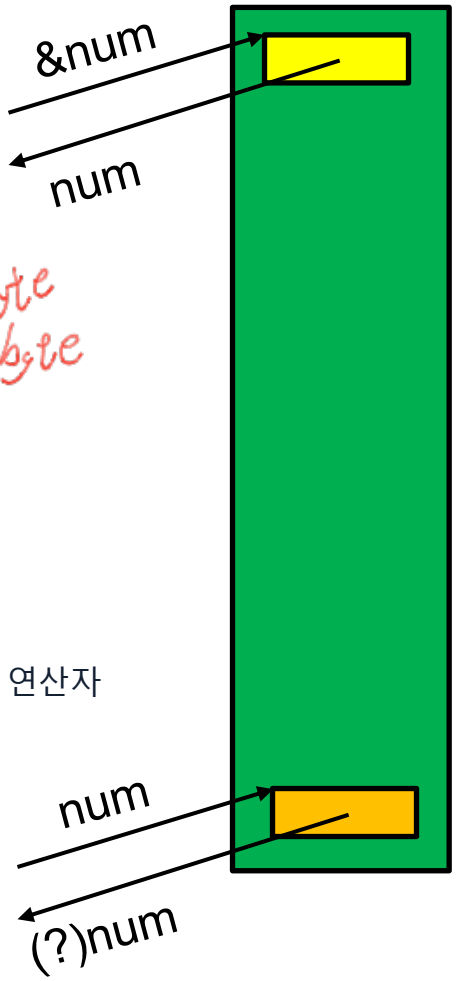
  - Ex) int num;
    - 1) 정수형으로 사용할 4 bytes 공간을 메모리에 확보해줘.
    - 2) 그리고 나중에 num이라고 하면 그 값을 나에게 가져와줘

  - 2) 의 개념이 매우 중요하다.

int = 4byte
char = 1byte

  - scanf("%d", &num); 이라고 명령을 내릴 시
    - Num은 num안에 저장된 값을 가져오는 개념이므로
    - 사용자가 입력한 정수를 num에 저장하는 개념과는 서로 호응하지 않는 명령이다.

    - 따라서, num으로부터 값을 가져오는 것이 아닌 num에게 값을 가져다 주는 방향 전환 연산자
    - & (ampersand, 앰퍼샌드)를 사용하여 scanf("%d", &num); 과 같이 사용을 했다.



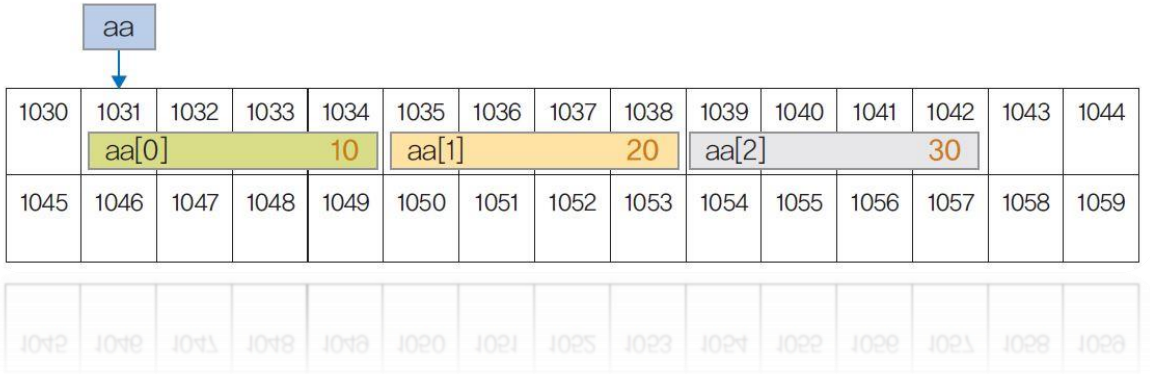
- **그렇다면,**
  - 애초에 num"에게"의 개념을 갖는 변수는 없나?
  - 그리고 어떤 다른 연산자를 써서 num"으로부터" 값을 가져오는 변수는 없는 걸까?

메모리의 구조와 변수를 위한 할당 위치 결정

● 배열의 메모리 할당

- **관전 포인트:** 화살표

```
int aa[3] = { 10, 20, 30 };
```



- 1) 배열의 이름만 사용하면?
- 2) 전체 값을 한 번에 확인 가능?

& aa[0]

● 배열의 메모리 표현

- 배열의 메모리 주소 표현
  - &aa[0] ➔ aa의 0번째 원소 " 에게 " ➔ 가리키는 지점의 실체 ➔ aa[0]의 주소 = 1031번지
  - aa[1]의 주소 (&aa[1]) = 1035번지
  - aa[2]의 주소 (&aa[2]) = 1039번지

int 이기 때문에 4byte

- 배열 이름 aa = 배열 a가 시작하는 지점의 주소 = 1031번지

  배열 aa의 주소를 구할 때는 '&'를 쓰지 않고 단순히 'aa'로 표현

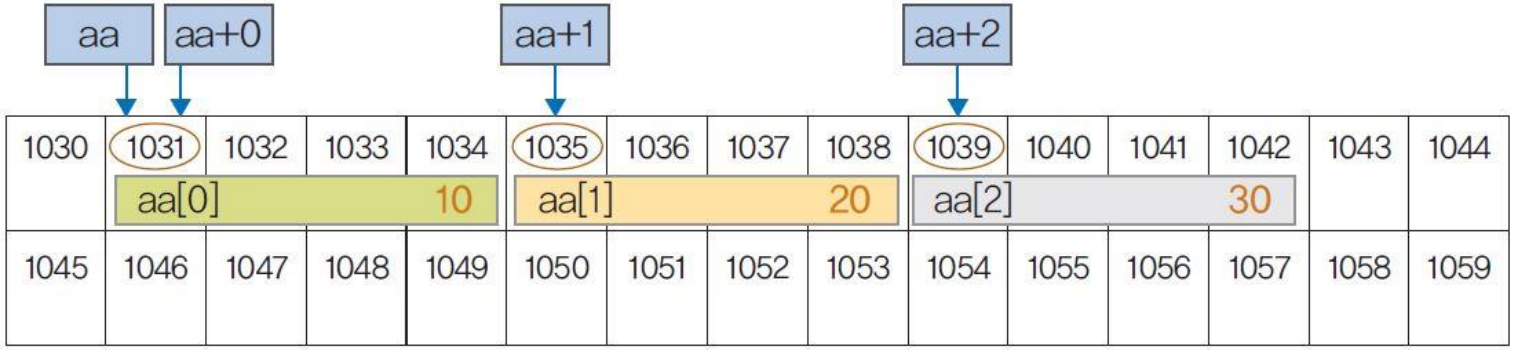
  aa라는 배열의 이름 자체가 애초에 "**에게**" 또는 "**으로**"의 개념이었다.

배열명의 활용법

- **aa 값을 1031로 가정하고 aa+1을 계산한 결과는 무엇일까?**
  - 예상 결과 : aa+1 → 1031 + 1 = 1032 (X)
  - 실제 결과 : aa+1 → 1031 + 4 = 1035 (O)
  - 이 1이 그 1이 아니야? 그럼 어떤 의미야? ➔ 원소 1개의 의미

- **주소 도출 과정**
  - '+1'의 의미 : 배열 aa의 위치에서 한 칸 건너뜀
  - 한 칸 : aa가 정수형 배열이므로 4바이트
  - 즉, aa+1 = &aa[1] = 1035

배열 첨자로 표현	배열 이름으로 표현	실제 주소
&aa[0]	aa + 0	1031
&aa[1]	aa + 1	1035
&aa[2]	aa + 2	1039



2 포인터 변수

포인터 변수의 개념

- **변수**
  - 어떤 값을 저장하는 메모리의 특정 공간

- **일반 변수**
  - 변수명으로 사용 시, 메모리 공간에 저장된 값을 미리 정의한 형식으로 읽어서 가져옴
    - int이면 정수형으로 읽고,
    - float이면 부동소수점 실수로,
    - char이면 문자형으로 읽어서 가져온다.

- **포인터 변수**
  - 변수명으로 사용시, 메모리 공간에 저장된 값이 다른 공간을 가리키는 메모리의 주소를 의미
  - 포인터 선언 : * 를 붙이고, 이름을 p로 시작하게 한다.
    - int* pnum; ➔ 어떤 정수형의 숫자를 가리키는 포인터 변수
  - 추가 의미: pnum이 가리키는 곳"으로부터" 값을 가져올 때는 *pnum의 형식으로 사용한다.
    - 포인터 변수의 방향 전환 연산자: *

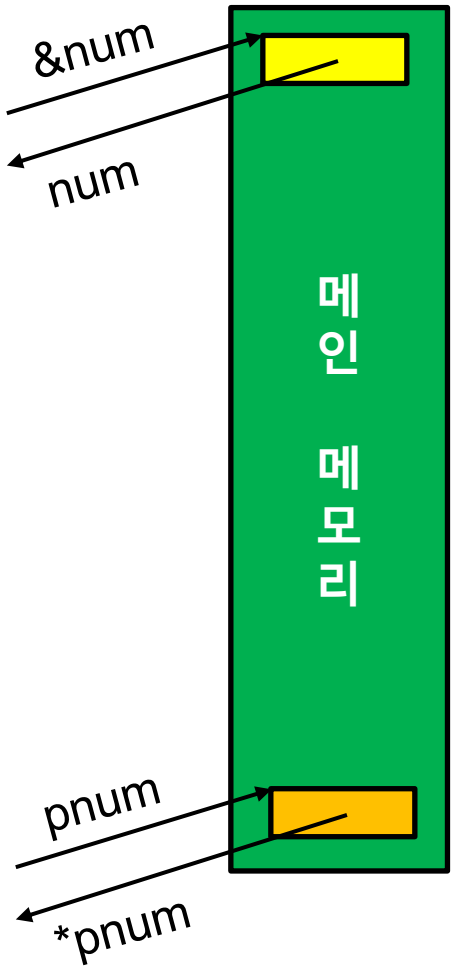
2 포인터 변수

포인터 변수의 개념

- **일반 변수와 포인터 변수의 실행 결과**

❶ char ch;	실행 결과 ▶	문자형 변수를 선언한다.
❷ char* p;	실행 결과 ▶	문자형 포인터 변수를 선언한다.
❸ ch = 'A'	실행 결과 ▶	문자형 변수에 문자 'A'를 대입한다.
❹ p = &ch;	실행 결과 ▶	포인터 변수에 변수 ch의 주소인 '&ch'를 대입한다.

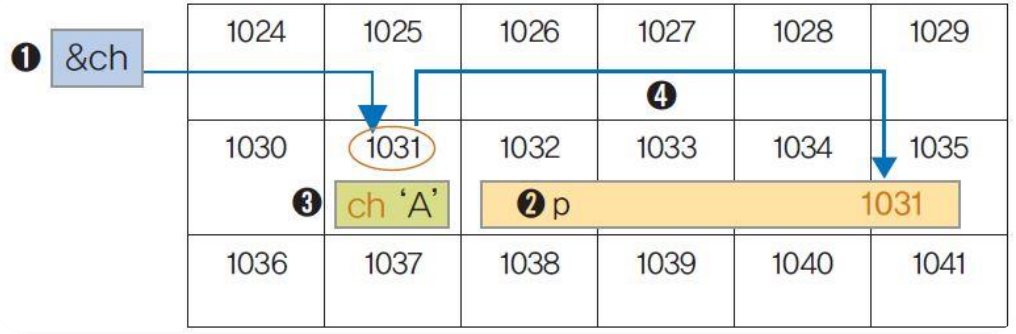
4byte



- **일반 변수와 포인터 변수의 방향 차**

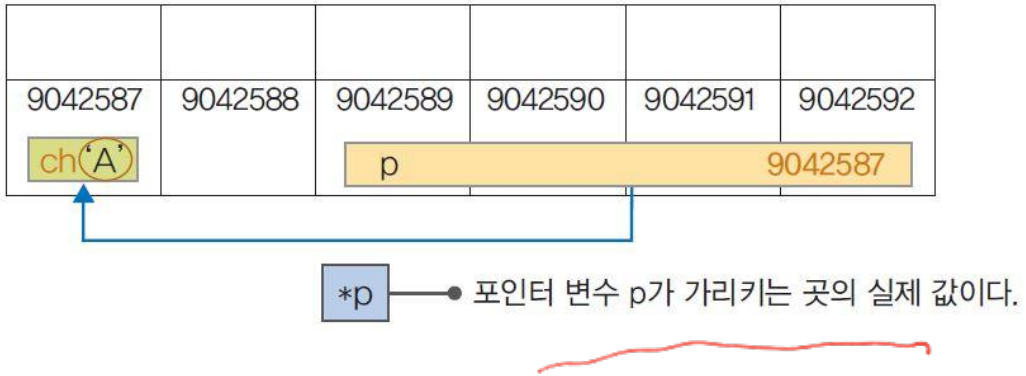
*p = 'A'

- 서로 역방향



포인터 변수의 개념

- **포인터 변수에 저장된 내용물의 실체**
  - 주소



- **일반 변수에 사용한 & 변수의 실체**
  - 주소

포인터 변수의 선언

❶ 포인터 변수 선언 방법 : 변수형 *

- 정수형 : int*, 문자형 : char*, 실수형 : float*

  일부 강의에서는 변수명 앞에 *를 붙여 int *pnum과 같이 사용하기도 하나

  본 강의에서는 자료형 뒤에 반드시 *를 붙이도록 함. ➔ int* pnum; char* pch; float* preal; 등

❷ char* p; 선언 시 p에는 다른 문자형 변수를 가리키도록 해야 함

- int* p; 선언 시 p에는 정수형 변수의 주소를 저장해야 함.
- 즉, 참조가 될 만한 같은 자료형의 다른 변수가 미리 만들어져 있어야 함.

int a;	실행 결과 ▶	정수형 변수 a를 선언한다.
char* p;	실행 결과 ▶	문자형 포인터 변수 p를 선언한다.
p = &a;	실행 결과 ▶	문자형 포인터 변수에 정수형 변수의 주소를 넣는다. (×)

⬇

int a;	실행 결과 ▶	정수형 변수 a를 선언한다.
int* p;	실행 결과 ▶	정수형 포인터 변수 p를 선언한다.
p = &a;	실행 결과 ▶	정수형 포인터 변수에 정수형 변수의 주소를 넣는다. (○)

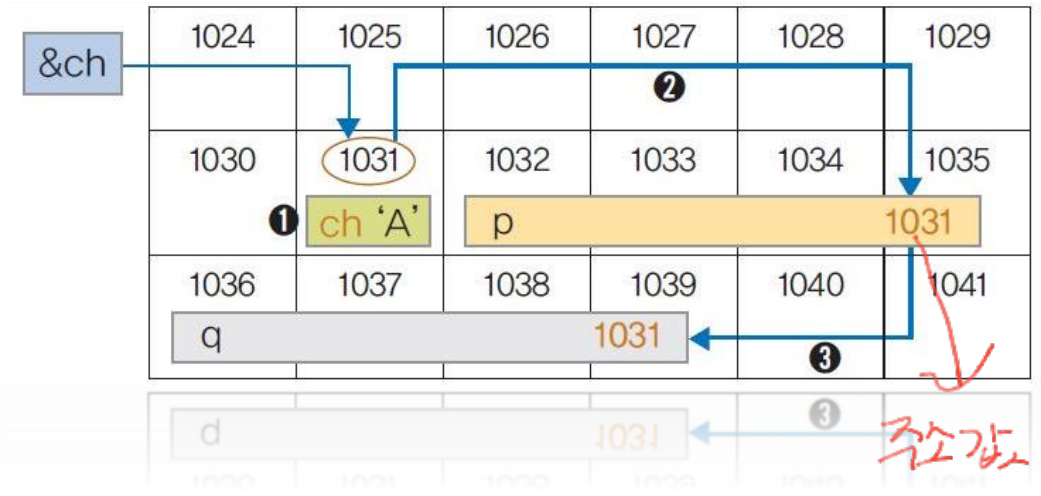
2 포인터 변수

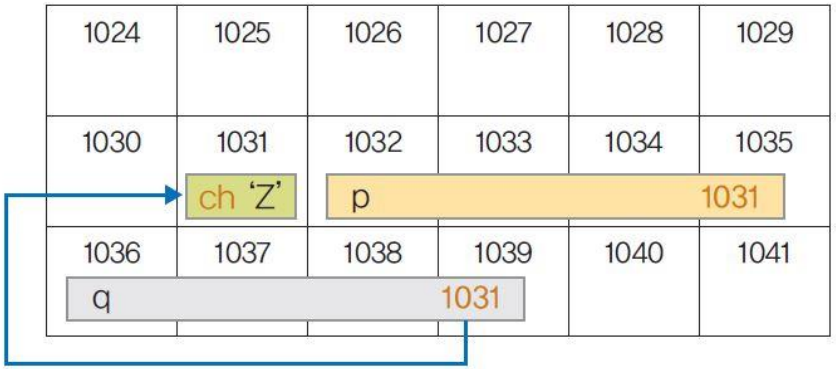
포인터 변수의 선언

- **다음과 같은 연산이 기능하려면 어떻게 코드를 작성해야 할까?**
  - 아래에 메모해보자.

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

char* q;
q = p;
```





*q — 포인터 변수 q가 가리키는 곳의 실제 값이다.

```
*q = 'Z';
```

문자형 배열과 포인터

문자열과 포인터

- 문자 하나를 저장하는 변수형 : char

- 문자 12개로 이루어진 문자열을 표현하는 변수의 선언: char s[12];
  - cf) 순수하게 개별 문자 12개를 다른 의도로 사용하는 문자형 배열: char[12] s;

I	T	-	C	o	o	K	B	o	o	k	\0

- char* pch = str : 배열 s와 포인터 p 호환 가능

  & str[1] = (pch+1)

  - str[3] == *(pch+3)

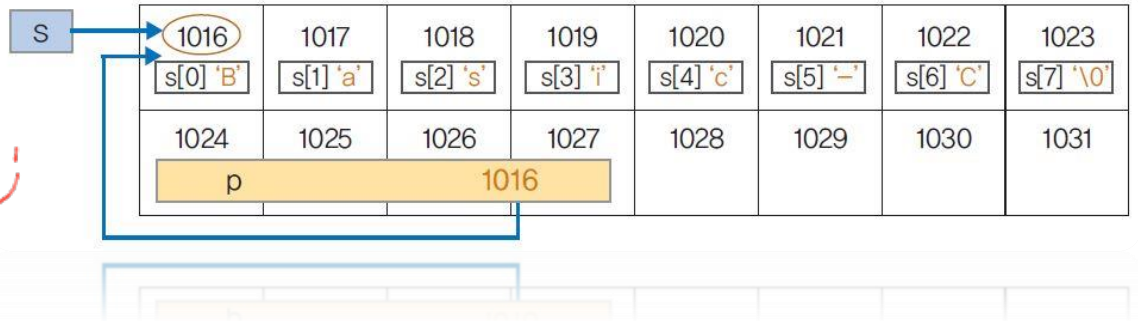
    str[1] == *(pch+1)

  - 변수명의 의미: 문자열(str)에서 그 중 문자(ch) 하나를 가리키는 포인터 변수(p)

문자형 배열의 연산과 포인터 변수의 크기

- **다음의 연산이 기능하도록**
  - 코드를 아래에 작성해보자.

```
char s[8]="";
char* p=s;
```



- **잠깐만**
  - char 는 크기가 얼마?
  - char* 는 크기가 얼마?

포인터 변수와 포인터 상수

- **아래 코드를 보고 물음에 답해보자.**

```
char s[3];
char* p;
p = s;
```

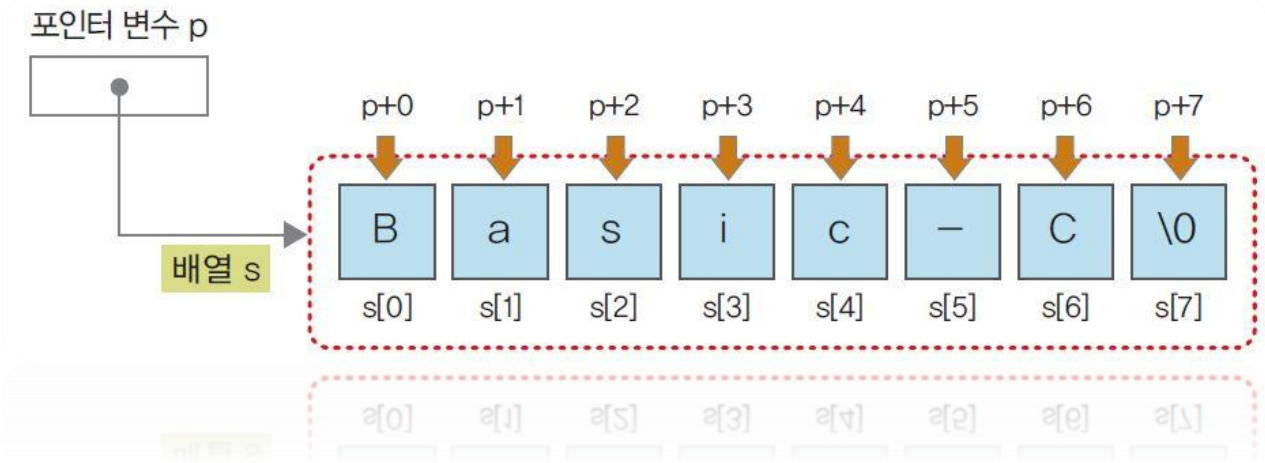
배열 첨자	포인터 상수	포인터 변수
s[0]	*(s+0)	*(p+0)
s[1]	*(s+1)	*(p+1)
s[2]	*(s+2)	*(p+2)

- 배열의 시작 주소 s는 다른 값으로 변할 수 있는가?

- 문자형의 포인터 변수 p는 다른 값으로 변할 수 있는가?

문자열 배열과 포인터의 응용

- 그림을 보고 의미와 기능을 파악해보자.



포인터 학습 노하우

❶ 포인터 변수가 무엇을 가리키는지 확인한다.

❷ 포인터 변수를 선언할 때는 자료형 바로 뒤에 *를 붙이고 변수 명 앞에 p를 붙이면 된다.

- ex) int* pnum; double* preal; char* pch;

❸ 포인터 변수에는 꼭 주소값을 넣어야 한다.

- 변수 이름 앞에 '&'를 붙임
- 배열의 이름은 그 자체가 주소이다.

❹ 포인터가 가리키는 곳의 실제 값을 구하려면 *를 붙인다.

- 포인터 변수 pnum는 변수 num이 있는 주소인 1016번지를 가리킨다고 가정함.

printf("%d", p)	실행 결과 ▶	a의 주소인 1016이 출력된다(주소 자체는 중요하지 않다).
printf("%d", *p)	실행 결과 ▶	p가 가리키는 곳의 실제 값, 즉 a 값인 123이 출력된다.

Summary

너 C 언어의 꽃이 누군지 아니?

0. 일반 변수와 포인터 변수

❶ 가지고 있는 방향성이 정 반대.

프로그램 실행 or 주소값 | 데이터

1. 메모리와 주소

❶ 프로그램에서 사용된 변수, 배열 등은 모두 메모리(램)에 존재하며 이 메모리의 각 자리에는 주소가 할당되어 있다.

❷ 변수의 주소를 표시하려면 변수 이름 앞에 &를 붙인다. &a

❸ 배열의 이름 자체는 배열의 시작 주소이다.

❹ 정수형 변수는 4바이트를, 문자형 변수는 1바이트를 차지한다.

2. 포인터

❶ 포인터 변수란 주소를 저장하는 변수로, 포인터 변수를 선언할 때는 *를 붙인다.

❷ 포인터 변수에는 반드시 주소만 대입해야 한다.

❸ 포인터 변수의 크기는 정수, 실수, 문자형에 관계없이 무조건 4바이트이다.

❹ 가변적인 문자열을 저장할 때는 배열보다 포인터가 더 적당하다.

3. 포인터와 배열

❶ 배열의 이름 그 자체는 변화할 수 없는 포인터이자 상수이다. ➔ 포인터 상수