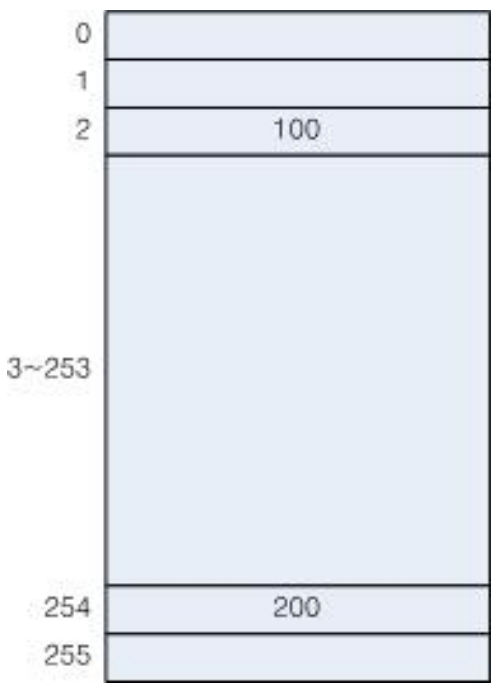
6. 포인터(pointer), [] 연산자

- 주소는 메모리의 각각의 셀cell에 붙여진 일련번호를 말합니다.



메모리의 주소: 그림에서 주소는 0~255입니다. 그러므로 주소를 나타내기 위해 8비트를 사용할 수 있습니다. 2번지에는 100이란 값이 들어 있습니다. 254번지에는 200이 들어 있습니다. 100이나 200을 정수로 생각해서는 안 됩니다. 100은 해석하기에 따라 정수, 실수 혹은 주소가 되기도 합니다. 우리가 int i=100; 이라고 코딩하면, 프로그래머의 입장에서는 i에 100이 대입되는 것으로 생각합니다. 하지만, 실제로는 컴파일러가 생성한 i의 위치 - 만약 그것이 2번지라고 하면 - 에 100이 들어가는 것입니다.

- 메모리 한 셀은 몇 바이트를 의미할까요?
- 1000번지는 2개의 바이트를 가리키는 것일까요, 아니면, 1바이트를 가리키는 것일까요?
- **바이트 접근가능 기계byte accessible machine**.
- C에서의 포인터 변수는 일반적인 '가리킨다(point)'라는 의미로 쓰인 포인터 변수와는 구분되어야 합니다.

포인터에 대한 이해

```
void main() {
    int i;
    int j=2;
    i=j;//이 문장에서 실제로 무엇이 일어나는가?
}
```

- 위의 프로그램은 j의 값을 i에 대입합니다.
- i=j라는 문장은 j를 i에 대입하는 것이 아닙니다.
- 또한 *j의 값*을 *i의 값*에 대입하는 것이 아닙니다.
- *j의 값*을 *i*에 대입하는 것입니다.
- 즉 등호(=)의 왼쪽에 쓰인 i와 오른쪽에 쓰인 j를 해석하는 방법이 다른 것입니다.

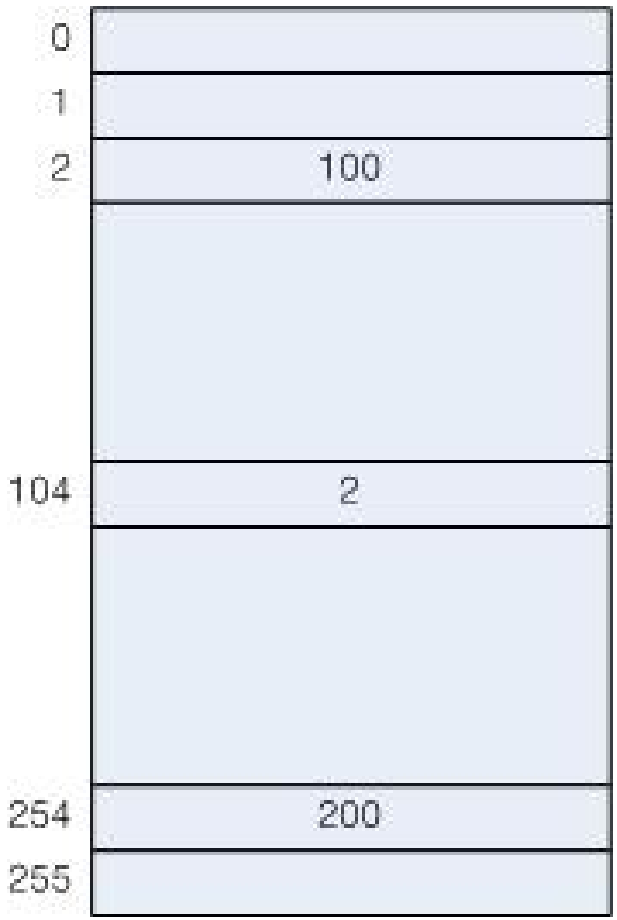
• 심벌 테이블(symbol table)

main()안에서는 두 개의 정수형 변수가 선언되었습니다. 컴파일러가 계산한 i,j의 메모리 주소(memory address)가 각각 100, 104였다고 가정합시다.

변수/함수 이름	실제 주소	형
i	100	int
j	104	int

심벌 테이블: 실제의 심벌 테이블에는 속성(attribute)이 3개만이 아닙니다. 지금은 설명을 위해 3개의 속성만을 나타내었습니다.

• j=2; 라는 문장에 의해 104번지에서 시작하는 4바이트에 2라는 정수값이 들어갑니다.



j=2; 의 실행후 메모리의 상태: [104]번지에 정수 값 2가 들어있습니다.

- 등호의 왼쪽 i는 심벌 테이블의 100을 의미합니다.
- 정수와 주소를 구분하기 위해 이를 [100]으로 쓰기로 합시다.
- 등호의 오른쪽 j는 심벌 테이블 [104]가 가리키는 값을 의미합니다.
- 이를 [104]*라고 쓰기로 합시다.

- 그러면 i=j; 는 아래와 같은 문장으로 번역됨을 알 수 있습니다.

[100]=[104]*

- 등호의 왼쪽에 오는 값은 반드시 주소address라야 합니다. 이를 **왼쪽 값(l-value, left value)**이라고 합니다.
- 등호의 오른쪽에 오는 값은 반드시 값value이어야 합니다. 이를 **오른쪽 값(r-value, right value)**이라고 합니다.

3=4;

- i의 주소값인 [100]을 메모리에 저장하는 방법은 없을까요?
- 정수형 주소변수는 다음과 같이 선언합니다.

```
int * ip;
```

*는 형 이름(int)과 명칭(identifier)사이에 위치합니다. 하지만, *는 구분자delimiter로서 하나의 토큰이므로, 위의 예에서 int와 혹은 ip와 흰공백(white space)으로 구분하지 않아도 됩니다. 즉, int* ip; int *ip; int * ip; 세가지 모두 좋습니다. C 스타일은 int *ip; 를 많이 사용했습니다. 하지만, C++ 스타일은 int* ip; 를 사용할 것을 권장합니다. 이것은 *가 형을 결정짓는 역할을 하기 때문에 그렇습니다. 하지만, *는 매 변수 이름마다 명시되어야 합니다. int *ip,i;는 ip를 포인터로 i를 정수로 선언한 것입니다. i도 포인터로 선언하기 위해서는 int *ip,*i; 처럼 선언해야 합니다. 그러므로 C++에서는 ip와 i를 포인터로 선언하는 방법으로 int* ip; int* i;를 권장하고 있습니다.

• int* ip;와 char * ip; 와의 차이점은 무엇일까요?
• 하나의 변수 선언 문장으로, 여러 개의 포인터 변수를 선언하기 위해서는 변수 이름마다 *를 붙여주어야 합니다.

```
int *ip, *ip2;
```

• 변수가 주소형 변수로 선언되지 않았을 때, 실제로 이 변수가 위치하는 메모리의 위치를 알 필요가 있습니다. 이것은 **주소 연산자(address-of operator, &)**로 가능합니다.

- &i는 [100]을 의미합니다.

• i=&j; 라는 문장은 가능하지 않습니다. 정수형 변수 i에 주소값을 대입하는 것은 가능하지 않기 때문입니다.
• **간접 지정 연산자indirect operator**

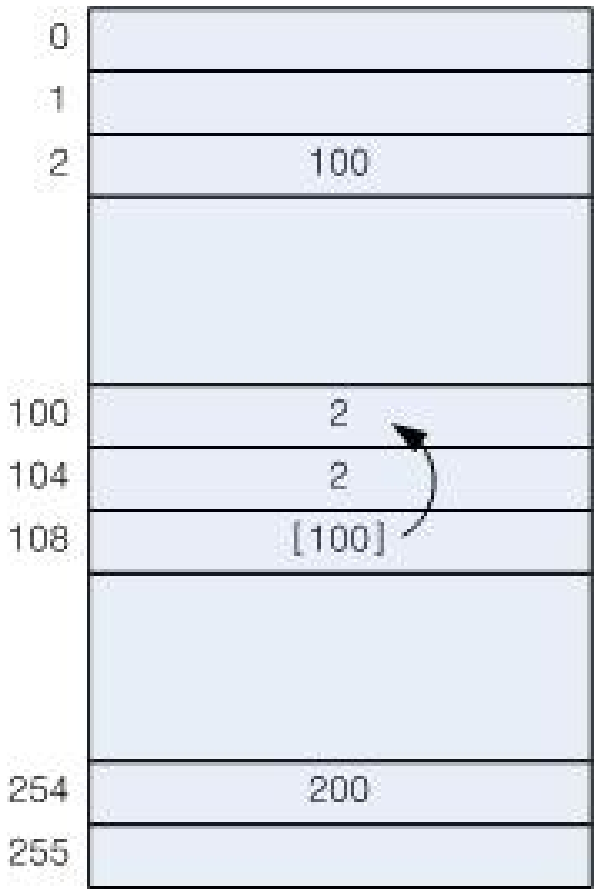
```
#include <stdio.h>

void main() {
    int i;
    int j=2;
    int* ip;//*는 ip가 포인터 변수임을 의미한다

    i=j;//이 문장에서 실제로 무엇이 일어나는가?
    ip=&i;
    printf("%d,%d,%p,%p\n",j,*ip,ip,&ip);//*는 간접지정연산자이다
}
```

변수/함수 이름	실제 주소	형
i	100	int
j	104	int
ip	108	int*

- 메모리의 구조는 다음과 같습니다.



문자열(string)은 포인터 표현.

- C는 문자열을 포인터로 관리합니다.
- 문자열 끝에 **문자열의 끝(end of string, EOS)**을 나타내는 특수문자 0이 위치합니다.

"hello\0world\0"는 "hello\0"와 같은 표현인가요?

그렇지 않습니다. "hello\0world\0"는 메모리를 13바이트(5+\0+5+\0+\0) 차지할 것입니다. 하지만, "hello\0"는 메모리를 7바이트 차지합니다. 그렇지만, 2개의 문자열을 출력하면 모두 hello를 출력할 것입니다. 왜냐하면 표준 출력함수는 '\0'을 만나면 문자열의 끝으로 판단하기 때문입니다.

• s의 형은 무엇이 되어야 할까요?

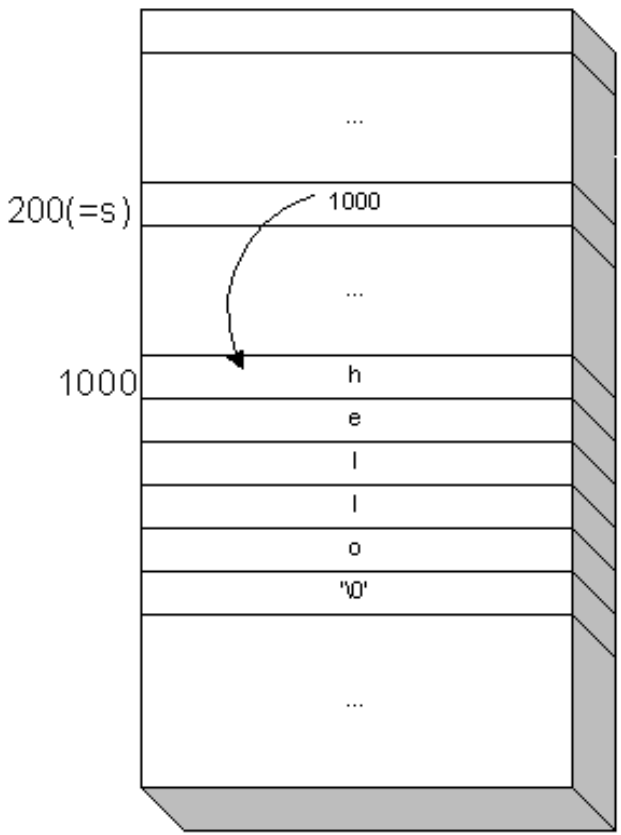
s="hello";

• s는 다음과 같이 선언되어야 합니다.

char *s;

• 메모리의 적절한 영역 - 컴파일러가 관리하는 힙(heap) - 에 "hello"+'₩0'을 차례대로 집어넣습니다. 문자 5개와 0(EOS)을 포함하여 6바이트를 사용합니다. 그리고 첫 번째 문자 'h'의 시작 주소를 돌려줍니다. 그러므로, "hello"는 'h'의 시작 주소 표현입니다. 그러므로 s는 char *s; 처럼 선언되어야 하는 것입니다.

"hello"+'\0'를 메모리의 적절한 영역에 집어 넣은 다음, 첫번째 문자 'h'의 시작 주소를 s에 대입합니다.



• *(s+1)은 무엇을 의미할까요? s가 1000번지이므로 [1001]*를 의미합니다. 즉 문자 'e'입니다. *(s+1)은 연산자 []를 사용하여, s[1]로 나타낼 수 있습니다.

$$*(s+n) \equiv s[n]$$

• s와 n의 위치를 바꾸어 *(n+s)라고 쓸 수 있듯이, n[s]라고 쓸 수 있음에 유의하세요.
• s[1]은 1[s]라고도 쓸 수 있습니다.

```
#include <stdio.h>

void main() {
    char *s;

    s="hello";
    printf("%c,%c,%c,%d\n",*(s+1),s[1],1[s],s[5]);
    //      e,e,e,0 0이 출력된다.
}
```

실습문제

1. 아래의 문장에 에러가 있다면 에러를 수정하세요(힌트: 에러가 아님).

```
#include <stdio.h>
void main() {
    char* s="hello"
            " world\n"
            "There was a"
            " white house in the town."
    printf(s);
}
```

---

2. 두개의 int* 타입의 변수 p와 q에 대해서 p[q]라고 사용하면 에러가 발생하는 이유에 대해서 설명하세요.

@