6차시 - 문자열(1)

문자열

- **사용 관점**
  - 아주 많이 사용되는 자료의 형태
  - 이름, 주소, 전화번호(-를 포함한) 등

- **C 프로그램 관점**
  - 문자(char)가 연속적으로 나열된 자료형(배열)
  - 포인터로 사용한다. (저장될 공간의 주소)

문자열

- **0개 이상의 문자로 이루어진 문자열**
  - 길이가 0인 문자열도 문자열이다.
  - 문자에는 알파벳 외에 '0'~'9'의 문자, 기호, 공백도 포함된다.
  - 각 문자는 ASCII 코드가 적용된다.

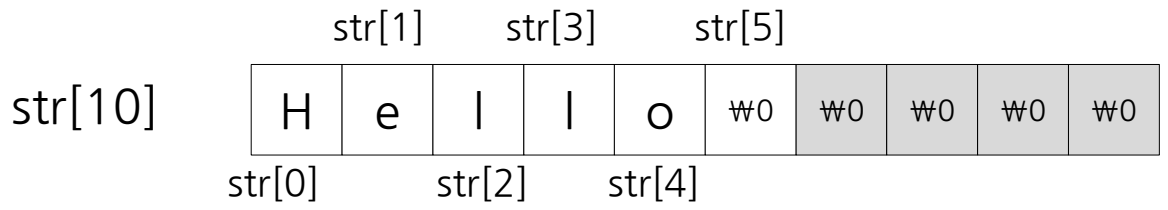
- **문자열의 적용**
  - 1개 이상의 char형 배열
  - 문자열 상수는 " " 사이에 기록
    - "hello", "Welcome", "" (빈 문자열)
    - 문자 상수는 ' ' 사이에 기록했다. 'A', 'c'

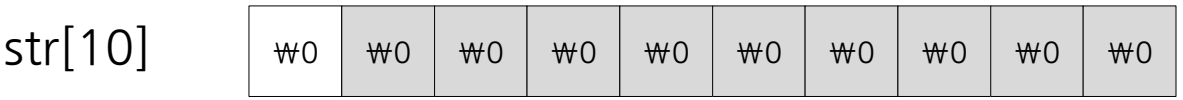
char c ; c = 'A';	char str[10]; str = "Hello"; //옳지 않으나 나중에 설명

문자열

```
char str[10] = “Hello”;
```

- **문자열로 10개의 공간을 할당**
  - 최대 9자 길이의 문자열을 저장 (n-1개의 문자열을 저장)
  - 문자열의 끝에는 무조건 ₩0(널문자, ASCII코드 0) 이 들어간다.





변수의 선언과 초기화

문자열의 기본 원칙

- 문자열 저장장소는 문자열 최대길이 + 1(널문자) 이다.
- 문자열도 변수의 선언과 함께 초기화할 수 있다.

```
char str[10];
char sentence[100]=“hello”;
```

변수의 초기화 (선언과 함께)

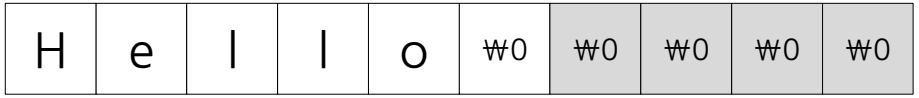
```
char str[10] = “alphabet”;

char sentence[] = “Hello World”;

char korean[10] = “안녕”;
```

변수의 선언과 초기화

변수의 초기화

```
char str[10];
```


```
char sentence[10]="hello";
```



- 초기화하지 않으면 쓰레기값
- 초기화한 바깥쪽 영역은 ₩0

} 배열에서 이미 배운 것

변수 선언과 별도로 문자열 변수에 값을 넣을 때에는 strcpy 함수를 이용해야 한다.

- `str = "Hello";` // 잘못된 문장이다. 오류메시지는?
- `strcpy(str, "Hello");` // 맞는 문장

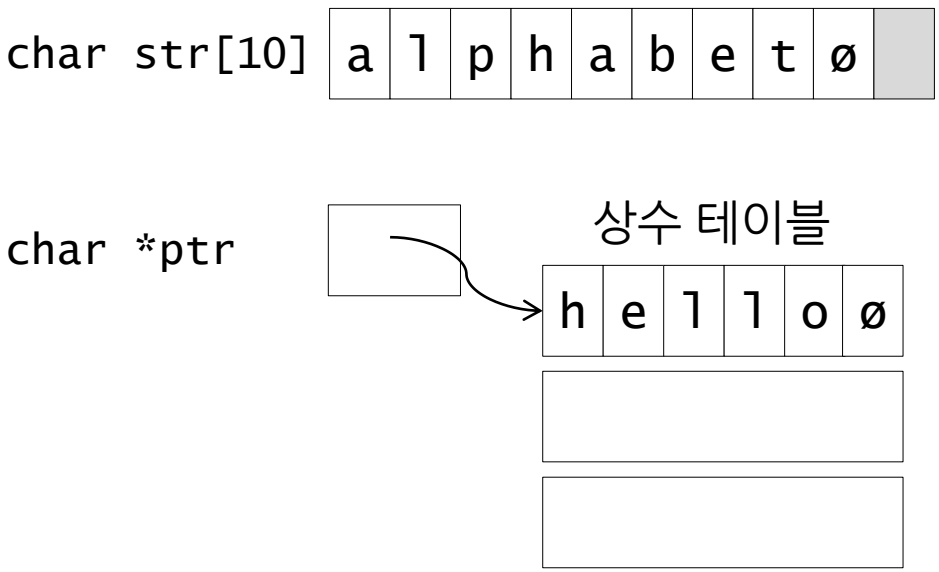
포인터와 문자열

포인터로 문자열을 선언한다.

```
char str[10];     // str 에 가면 문자열이 있다. 문자열 저장 장소
char *p;          // p 주소에 가면 문자열이 있다. 문자열이 있는 곳의 포인터
```

포인터 선언과 초기화

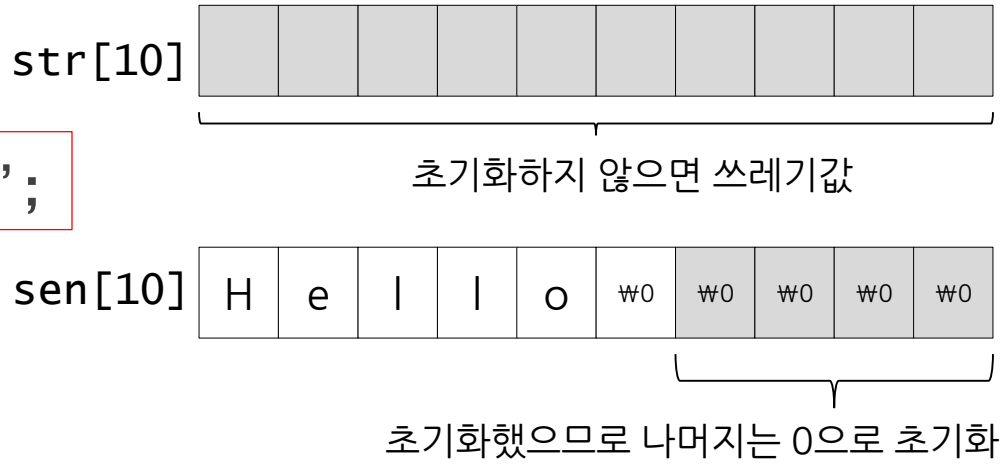
```
char str[10] = "alphabet";
char *ptr = "hello";
```

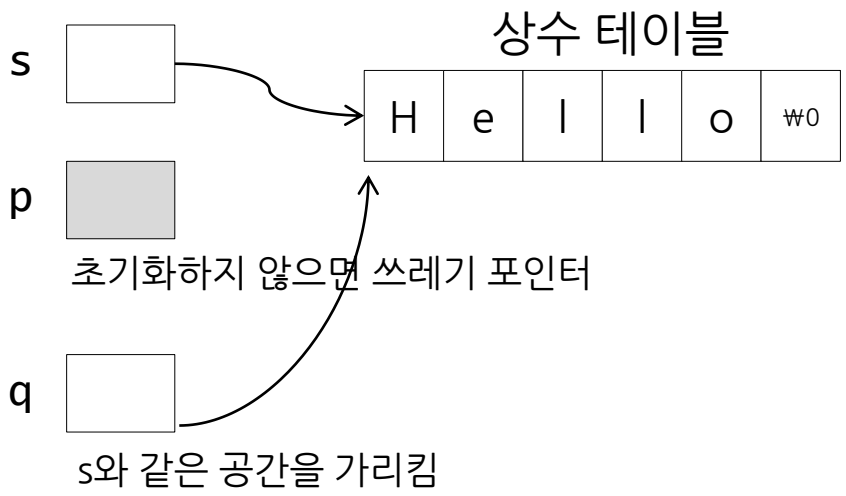


포인터와 문자열

```
char str[10];
char sen[10] = "Hello";
char *s = "Hello";
char *p;
char *q = s;
```

두 개는 완전히 다르다.





포인터와 문자열

```
char sen[10] = “Hello”;
sen[0] = ‘h’;     // 문제 없음

char *str = “Hello”;
str[0] = ‘h’;

sen = “World”;    // 잘못된 문장
str = “World”;    // 문제 없음
```

그림을 그려보자.

포인터와 문자열

```
char   *p, *q;
char   sen[ ] = “Hello ”;
char   *s = “World”;

printf(“%s %s\n”, sen, s);
p = sen;
q = s;
printf(“%s %s\n”, p, sen);  // 출력 결과는 같다.
printf(“%s %s\n”, q, s);    // 출력 결과는 같다.
```

문자열 배열

- char str[3][10] = {“Hello”, “World”, “Program”};
  - strcpy(str[0], “Hello”);
- char *pstr[3] = {“Hello”, “World”, ”Program”};

11.2 문자열의 입출력

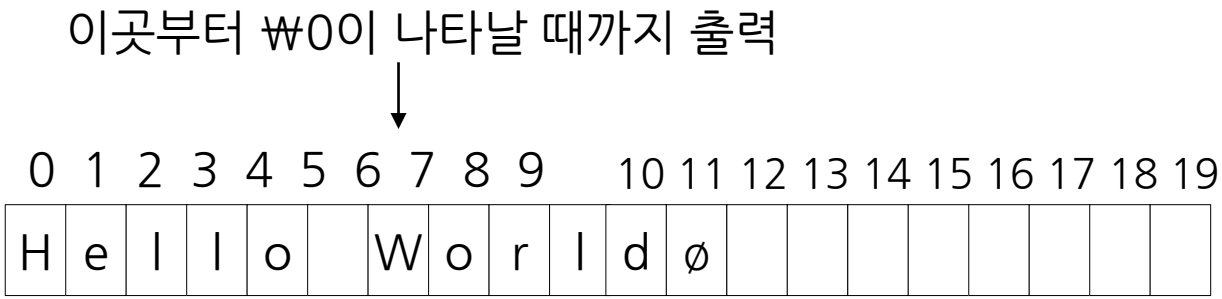
출력

- printf(“%s”)를 이용한 출력

```
char  str[ ] = “Hello World”;
printf(“%s\n”, str );

printf(“%s\n”, &str[0]);
```

- 포인터 위치만 바꾸면 일부분의 출력도 가능하다.

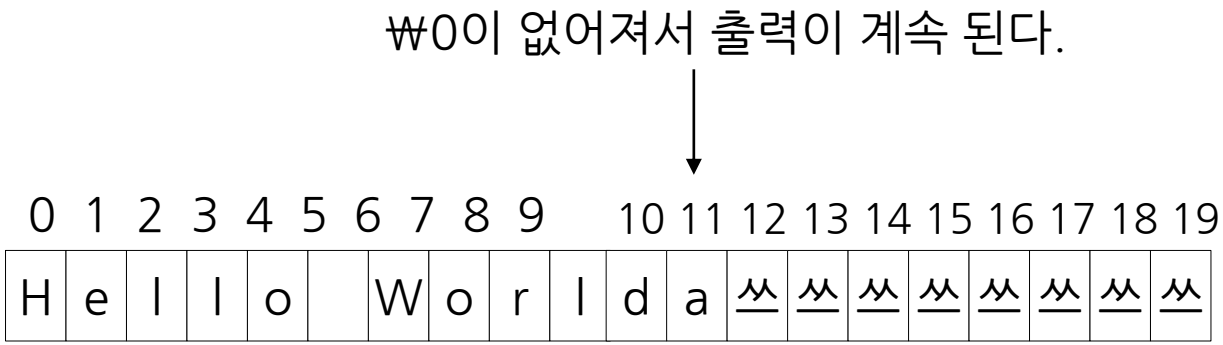
```
printf(“%s\n”, &str[6]);   // World 만 출력된다.
                           // 출력 시작 위치를 바꾼 것이다.
printf(“%s\n”, str+6 );    // 위와 같다.
```



출력

- ‘₩0’이 없어지면 쓰레기가 출력된다.

```
str[11] = ‘a’;                  // ‘\0’을 지웠다
printf(“%s\n”, str);            // 쓰레기 출력. 메모리 문제도 발생 가능성
```



입력

- `scanf("%s", str);`
  - str 이 포인터이므로 &를 붙이지 않았다.
  - `scanf("%s", &str[0]);` 과 동일
  - `scanf("%s", str + 5);` 와 같이 **지정된 위치에 입력도 가능**.
  - scanf는 공백을 구분자로 사용하여 단어만 입력된다.
    (문장 입력이 안 된다)
  - 메모리 문제를 유발할 수 있다. (공간 부족)
    - scanf는 공간을 확인하지 않는다.

입력

- `char str[10];`
- `scanf("%s", str+1);  // 여기서 "abcd"를 입력`
- `printf("%s\n", str);`
- **결과는?**

scanf, scanf_s

- char str[10];
- scanf(“%s”, str); // 10문자를 넣으면?
- scanf(“%s”, str, 10); // 10문자를 넣으면?
- scanf(“%9s”, str); // 10문자를 넣으면?

다른 입출력 함수들

- `char *gets ( char *str);  // 입력용`
  - 한 행 전체를 입력(공백문자 포함)
  - 빈 문자열(“”) 입력도 가능 (scanf는 불가능)
  - ‘₩n’은 제거됨

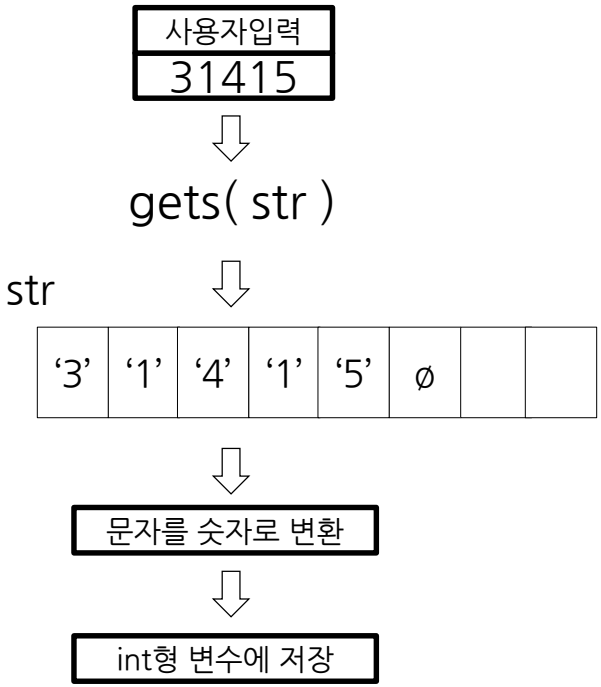
- `int  puts(const char *str);  // 출력용`
  - 양식 지정 없음.
  - 출력된 글자 수 반환

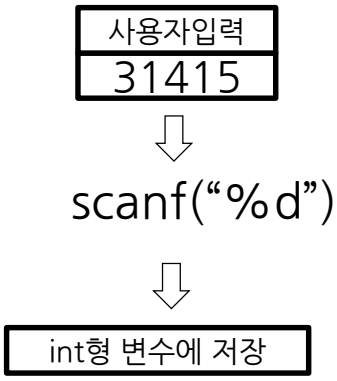
입력 방식 혼용

```
int  i;
char  str[10];
scanf("%d", &i);
gets(str);
```

- **str는 입력 받지 않고 지나간다. ""이 들어간다.**
  - 입력 버퍼의 처리 문제
  - scanf("%c")에서도 발생했었다.
  - fflush로 해결했었다.
- **대책**
  - fflush(표준이 아님)나 rewind 사용
  - scanf와 gets를 혼용하지 않는다. 두 함수의 방식이 다르다.
  - 줄 바꿈 제거를 위한 gets를 한 줄 더 넣는다.
    - getchar() 을 넣을 수도 있다.

scanf vs gets






```
// 과정은 복잡하지만 다양한 체크 기능 추가가 가능
gets(str);
while( ( str[index]) >= '0') && ( str[index]) <= '9') ) {
    // 문자를 숫자로 변환하여 i 에 저장
     i = i * 10 + istr[index] - '0';
    index++;
}
```

범위 초과 대비

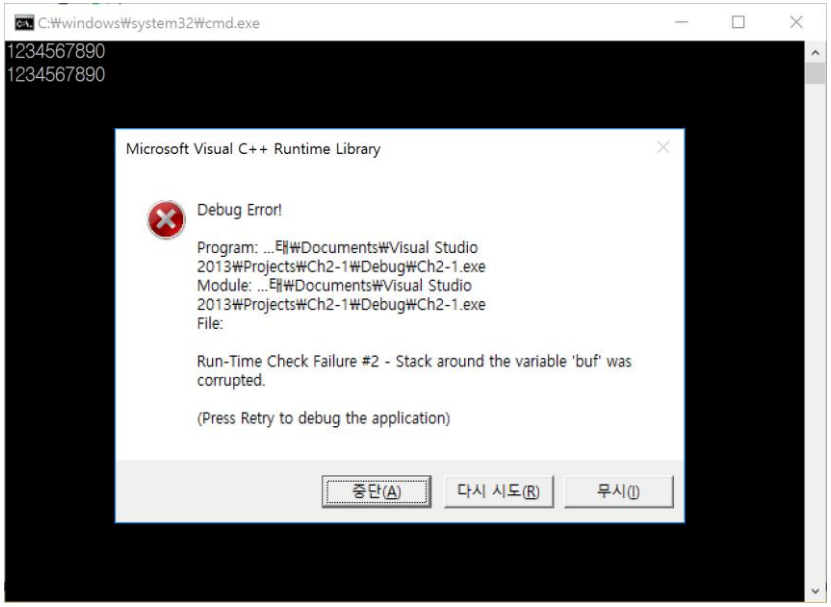
- **문자열 변수의 크기를 초과한 입력은 프로그램 비정상 종료**
  - 변수는 10글자인데, 사용자가 30자 입력함
- **gets에서 입력 변수의 범위를 초과할 수 있다.**
  - `gets_s` (Microsoft의 해결법. 표준이 아니다)
  - `char *gets_s(char *Buf, rsize_t Size);  // 크기를 감지한다.`

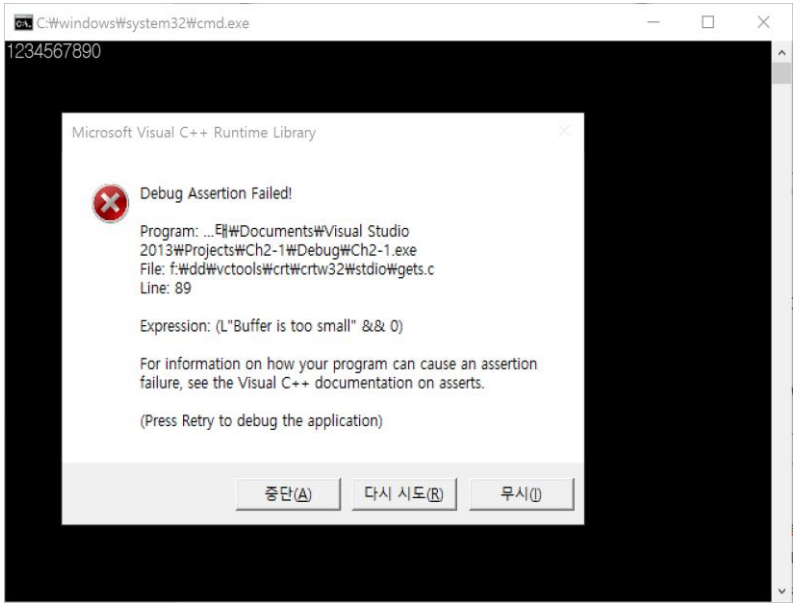
```
int main(void)
{
        char buf[10];

        gets(buf);
        printf("%s₩n", buf);
}
```

```
int main(void)
{
        char buf[10];

        gets_s(buf, sizeof(buf));
        printf("%s₩n", buf);
}
```

범위 초과 대비

- 두 결과의 차이는?





범위 초과 대비

- **fgets (파일 입출력 함수)**
  - 표준 입력에서 사용 가능
  - `char *fgets(char *Buf, int MaxCount, FILE *file)`
  - '\n' 문자가 남아있다.
  - `fgets(str, 9, stdin);  // 9글자로 제한한다.`

```
char buf1[6], buf2[6];

fgets(buf1, 6, stdin);
fgets(buf2, 6, stdin);
printf("%s : %s₩n", buf1, buf2);
```

⇦ "alphabet"

결과는?