

민 코 딩 훈 련 2 코 스

수업노트 LV-25



배우는 내용

1. 여러 값 추출하기
2. split
3. insert / erase 메서드
4. replace
5. 유효성 검사

여러 값 추출하기

단순 파싱 – for문을 쓰지 않는 방식

- ▶ [] 괄호 안에 있는 문자열 2개 추출하기
- ▶ 비슷한 코드가 2번 반복 됨

```
string str = "AB[KFC]HI[BBQ]HH";

int a = 0;
int b = 0;

a = str.find('[', 0);
b = str.find(']', a + 1);
int size = b - a - 1;
cout << str.substr(a + 1, size);

a = str.find('[', b + 1);
b = str.find(']', a + 1);
size = b - a - 1;
cout << str.substr(a + 1, size);
```

for를 사용한 파싱

▶ 반복문을 사용한 파싱

다음 검색을 위해,
검색 시작 index를
지정해준다.

```
for (int i = 0; i < 2; i++) {  
    a = str.find('[', a);  
    b = str.find(']', a + 1);  
    int size = b - a - 1;  
    cout << str.substr(a + 1, size);  
    a = b + 1;  
}
```

모든 [] 괄호 안, 단어 파싱

- ▶ while문을 사용한 파싱

```
string str = "AB[KFC]HI[BBQ]HH[WORLD]HOHO";
```

```
int a = 0;
```

```
int b = 0;
```

```
while(1) {
```

```
    a = str.find('[', a);  
    if (a == -1) break;  
    b = str.find(']', a + 1);
```

시작과
끝 지점 선택

```
    int size = b - a - 1;  
    cout << str.substr(a + 1, size);
```

```
    a = b + 1;
```

문자열 추출

```
}
```

다음 검색 시작 지점 지정

```
return 0;
```

The background features several overlapping, semi-transparent light blue circles of varying sizes. The word "split" is written in a bold, dark blue, sans-serif font, positioned on the left side of the image, partially overlapping one of the larger circles.

split

C++ 파싱의 단점 1

Split 메서드가 C++에는 없다.

- ▶ split : 특정 문자열을 기준으로 잘라, 배열로 만들어 주는 가능
- ▶ 이번 챕터에서는 split을 직접 만드는 연습을 한다.

"ABC|ERW|QQ|BGT"

"|" 를 기준으로
split 하기

ABC

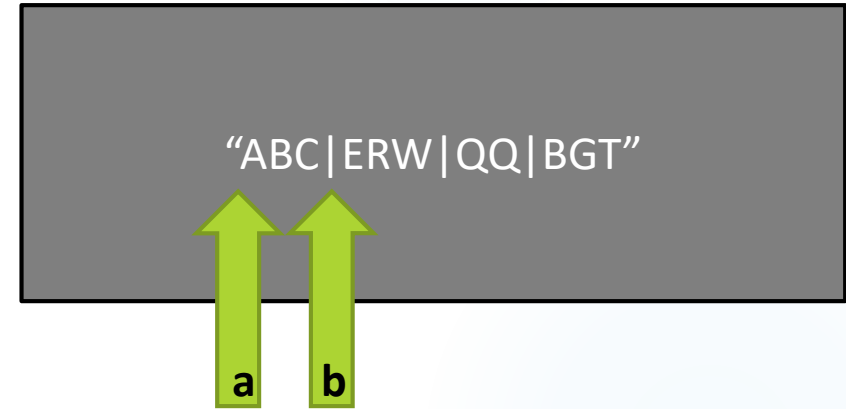
ERW

QQ

BGT

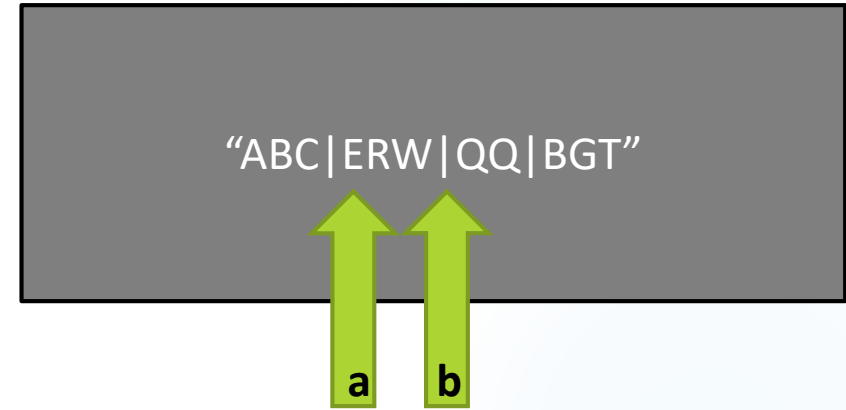
split 원리 1

1. $a = 0$
2. $b = \text{"|"} \text{ 지점을 선택한다.}$
3. $a - b$ 지점을 substring으로 자른다.
4. 다음 검색 시작 지점을 $b + 1$ 로 지정한다.



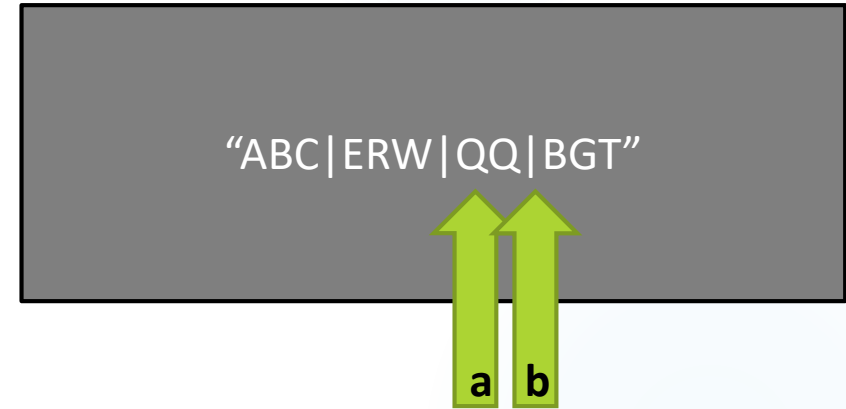
split 원리 2

1. $b = \text{"|"} \text{ 지점을 선택한다.}$
2. substring으로 자른다.
3. 다음 검색 시작 지점을 $b + 1$ 로 지정한다.



split 원리 3

1. $b = \text{"|"} \text{ 지점을 선택한다.}$
2. substring으로 자른다.
3. 다음 검색 시작 지점을 $b + 1$ 로 지정한다.

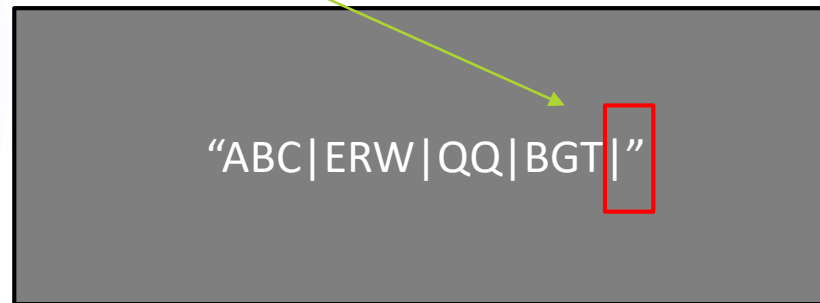
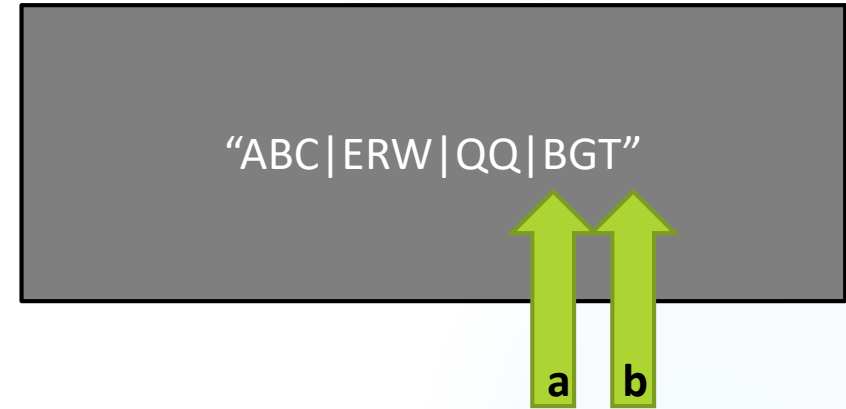


split 원리 4 – ISSUE!!

1. b = “|” 지점을 선택한다.

▶ 이 부분에서 마지막 위치를 찾지 못한다.

▶ 이러한 문제를 해결 하기 위해
맨 마지막에 | 를 하나 더 추가해 두고 파싱을 하면 된다.



split 구현하기

▶ while을 돌면서,

vect 배열에
파싱한 문자열을 하나씩 넣는다.

```
string str = "ABC|ERW|QQ|BGT";

//예외처리
str += "|";

int a = 0;
int b = 0;

string vect[10];
int t = 0;
while (1) {
    b = str.find("|", a);
    if (b == -1) break;

    int size = b - a;
    vect[t++] = str.substr(a, size);

    a = b + 1;
}
```

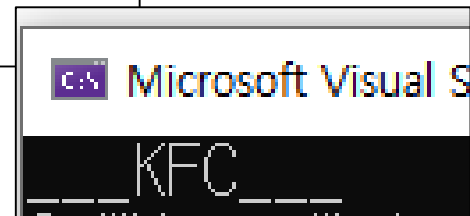
조사식 1	
검색(Ctrl+E)	
이름	값
▼ vect	0x0080f8cc
▶ [0]	"ABC"
▶ [1]	"ERW"
▶ [2]	"QQ"
▶ [3]	"BGT"
▶ [4]	" "

insert / erase

insert : 문자열 넣기

- ▶ 특정 index에
문자열을 집어 넣는다.

```
string str = "_____";  
str.insert(3, "KFC");  
cout << str;
```



erase : 문자열 삭제

- ▶ 특정 index에서 특정 글자 수 만큼 문자열을 제거한다.

```
string str = "ABC-----DEFG";  
  
str.erase(3, 5);  
  
cout << str;
```



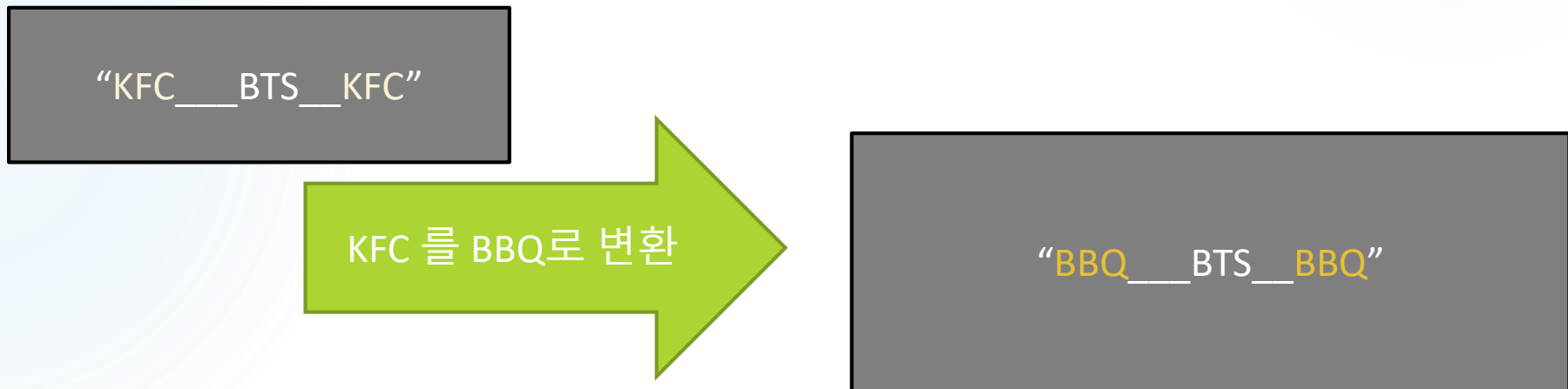
Microsoft Visual Studio
ABCDEFGG

replace

C++ 파싱의 단점 2

Replace 메서드가 C++에는 없다.

- ▶ repalce : 특정 문자열을 찾아, 다른 문자열로 교체를 해주는 메서드
- ▶ 직접 구현해주자!



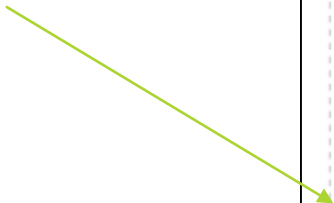
replace 과정

1. 변경할 문자열을 찾는다.
2. 문자열을 지운다. (erase 메서드)
3. 새로운 문자열을 추가한다. (insert 메서드)

relace 소스코드

- ▶ 문자열을 찾아 지우고, 추가한다.

다음 검색을 시작할
index 계산 방법



```
string str = "KFC__BTS__KFC";

string oldStr= "KFC";
string newStr = "BBQ";

int a = 0;
int b = 0;
while (1) {
    b = str.find(oldStr, a);
    if (b == -1) break;

    str.erase(b, oldStr.length());
    str.insert(b, newStr);

    a = b + newStr.length();
}
```

유효성 검사

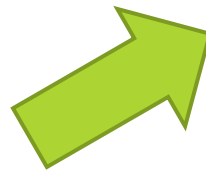
유효성검사

- ▶ 유효성검사 (valid check) 는 허용되는 값이 들어오는지 확인할 때 쓰인다.
- ▶ ID, Password가 정상적인 값인지, 아닌지 검사에 쓰인다.

LOGIN

로그인

☐ 아이디 저장 ☐ 자동 로그인



로그인



로그인

유효성 검사 문제

> 디버깅이 힘든 개발 방법

- ▶ ID 문자열 유효성 조건
 - ▶ 소문자가 없어야 함
 - ▶ 숫자가 하나 이상 존재

```
int main()
{
    string id = "ABC931";

    int flag1 = 0;
    int flag2 = 0;
    for (int i = 0; i < id.length(); i++) {
        if (id[i] >= 'a' && id[i] <= 'z') {
            flag1 = 1;
            break;
        }
    }

    if (flag1 == 0) {
        for (int i = 0; i < id.length(); i++) {
            if (id[i] >= '0' && id[i] <= '9') {
                flag2 = 1;
                break;
            }
        }

        if (flag2 == 1) cout << "PASS";
        else cout << "FAIL";
    }
    else cout << "FAIL";

    return 0;
}
```

유효성 검사 문제

> 디버깅이 편리한 개발 방법

유효성 검사 문제는
반드시 함수로 빼서, 구현하자!

- ▶ 탈락 조건들을 하나씩 구현한다.
- ▶ 탈락 조건들을 피해, 살아남았다면 1 리턴

```
int isValid(string id) {  
  
    //소문자면 탈락  
    for (int i = 0; i < id.length(); i++) {  
        if (id[i] >= 'a' && id[i] <= 'z') return 0;  
    }  
  
    //수가 존재하지 않으면 탈락  
    int flag = 0;  
    for (int i = 0; i < id.length(); i++) {  
        if (id[i] >= '0' && id[i] <= '9') {  
            flag = 1;  
            break;  
        }  
    }  
    if (flag == 1) return 0;  
  
    //여기까지 살아남았다면, 유효성 검사 성공  
    return 1;  
}  
  
int main()  
{  
    string id = "ABC931";  
  
    if (isValid(id)) cout << "PASS";  
    else cout << "FAIL";  
  
    return 0;  
}
```