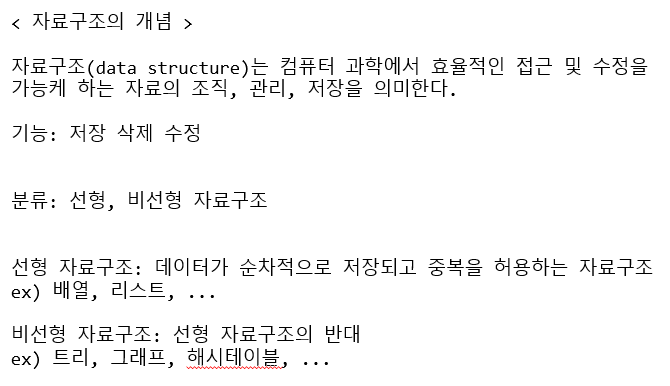
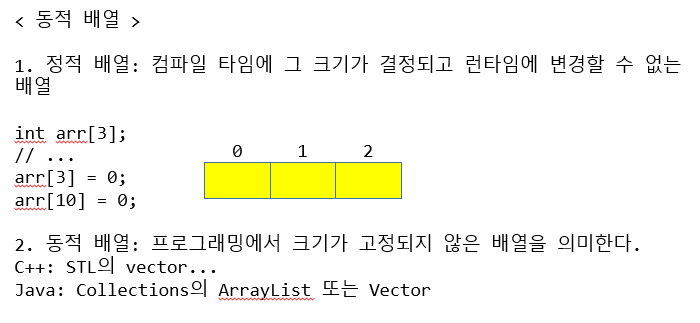
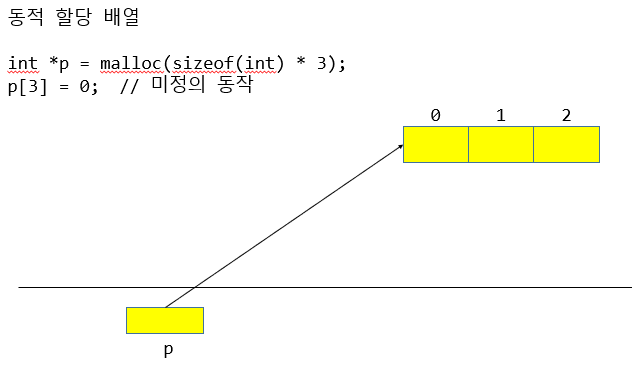
Day0527



[자료구조의 개념/종류]



[정적배열 / 동적배열]



[동적할당배열(동적배열과 동적할당배열은 다르다!)]

->동적할당배열은 배열이 할당될 때 크기가 고정되는 배열!!

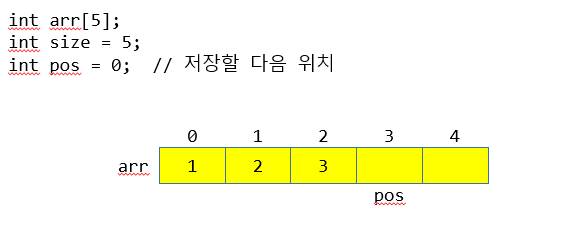
**<<<<<추가>>>>>**

🡺0527 1.c

배열의 한계

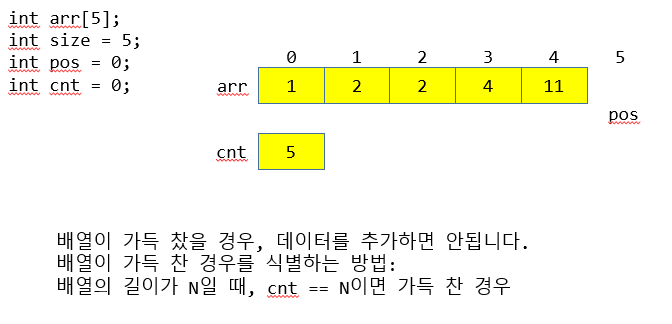
1. 배열의 크기(길이) 정보가 없음
2. 배열의 범위를 넘어설 경우를 알수 없음(ex>배열 5개 짜린데 10번방의 값을 바꾸려하면 런타임에 에러..)
3. 배열의 크기를 조절할 수 없음

cf) sizeof(\*arr) –> 크기4 / sizeof(arr) –> 크기20



1.정수를 저장하는 배열을 생각해보자

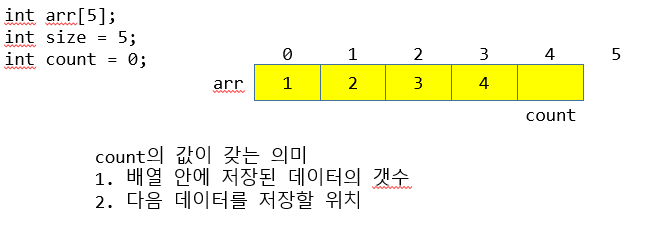
pos를 도입하면 순차적으로 배열을 저장할수있다.



그러나 배열이 가득찬 상황이면, 데이터를 추가하면 안된다.

배열이 가득 찬 경우를 식별하는 방법: 배열의 길이가 N일 때 cnt == N 이면 가득 찬 경우

그러나 cnt값과 pos값이 같아 버리므로 굳이 pos값이 필요가 없다.

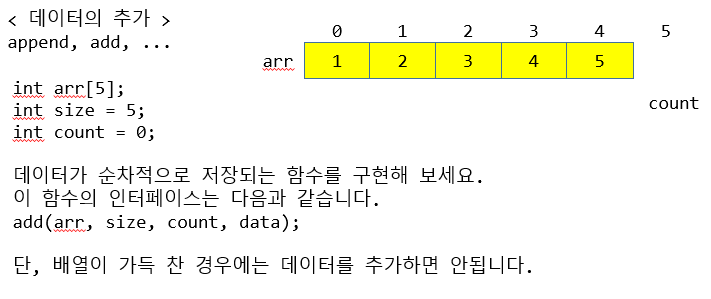


pos값을 없애고, cnt->count로 이름 바꿈

count의 값이 갖는 의미 1. 배열안에 저장된 데이터의 개수

2. 다음 데이터를 저장할 위치

🡺0527 2.c



<데이터의 추가>

2개 이상의 자료구조를 만들수없는 단점이 있다.

그 이유로는 자료구조의 연산을 하는 내부구조로 전역변수의 심볼을 사용하고있기 때문에!

그것을 해결하기 위해 함수에서 전역심볼을 사용하지 않으면된다.즉, 전역심볼을 지역심볼로 바꾸면 된다(아래코드)

🡺0527 3.c

2개 이상의 자료구조를 생성할수 없다는 문제가 해결됨(지역변수를 사용함수를 사용함으로서)

그러나 이 예제는 코드가 매우 복잡하다는 경향이 있다. 그래서 그것을 해결하기 위해 추상화를 해야한다. 즉, 구조체로 만들어서 그 단점을 극복해보자

🡺0527 4.c

구조체는 지역변수로 선언되어있으므로 꼭 Array구조체 안의 변수에 값을 넣어주자.

안넣게 되면 쓰레기값이 들어가게 된다.

구조체안의 값의 초기화는 자료구조가 해주는게 좋다.(그렇지 않으면 코드를 작성한 사람만 쓸수 있을것이다..) 그래서 아래 코드를 보면

🡺0527 5.c

구조체 안의 내용을 초기화를 하는 함수(자료구조)를 만들어주자.

그러나 이 코드는 자료구조의 컨텍스트를 저장하기 위해 스택 메모리를 사용하고 있다는 단점이 있다. 이를 해결하기 위해 정보 구조체를 힙에 생성하도록 하자

🡺0527 6.c

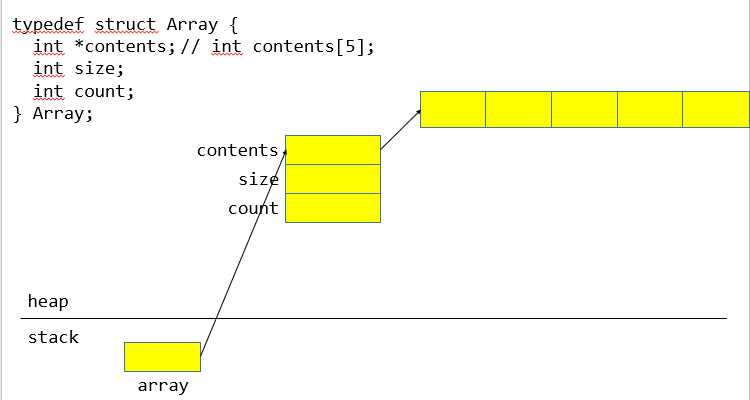
스택 메모리를 사용하지 않고 힙에 생성하는 예제

🡺0527 7.c

이전 예제는 함수로 전달되는 배열에 대하여 그 방식이 일반화 되어있지 않다. 어떨땐 \*\*로 넘기고 어떨땐 \* 로 넘기는등… (&arr, arr). 이를 해결한 코드

arrayCreate()에 굳이 매개변수를 넘길 필요없이 그냥 함수안에서 Array 만들고 그것을 반환하여 main에서의 Array \* arr에 넣어주면 된다.

🡺0527 8.c

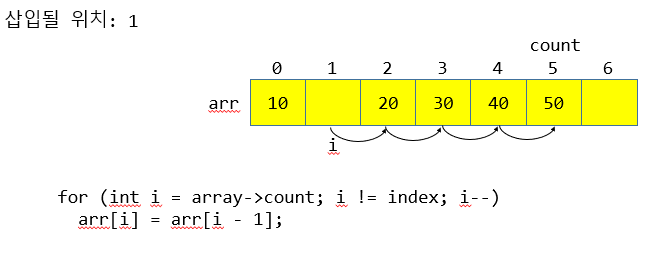
이전 자료구조는 배열의 크기가 고정되어 있다는 단점이 있다. 이를 해결하기 위해 contents에 포인터를 도입하여 동적으로 생성해준다.

**<<<<<수정>>>>>**

🡺0527 9.c

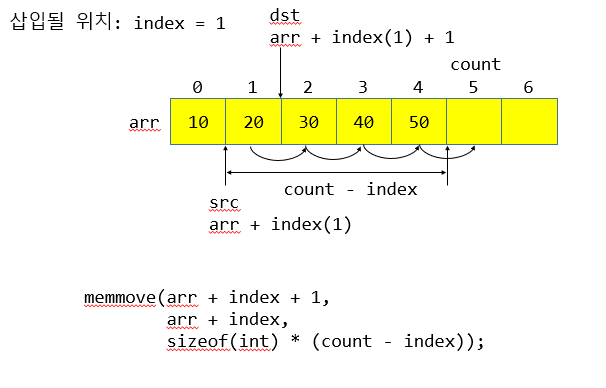
기존 배열에 특정 위치에서 새로운 데이터로 바꾸는 예제 -> arraySet

🡺0527 A.c

기존의 배열에 새로운 데이터를 삽입하는 예제 ->arrayInsert

memcpy는 절대 쓰면 안된다. 같은 메모리 상이므로! 중복되는곳이 있다면 큰일남

그럴때엔 memcpy가 아니라 memmove를 쓰자!!

memmove 예제

**<<<<<읽기>>>>>**

🡺0527 b.c

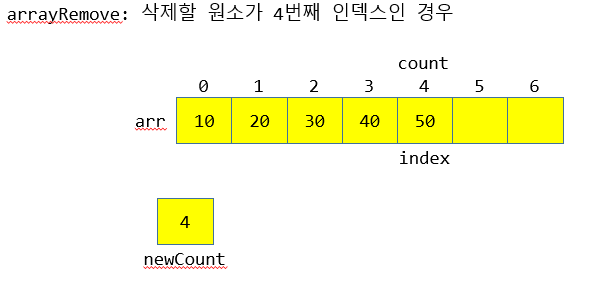
데이터를 읽어오는 함수를 추가하는 코드

arrayCount / arrayGet 추가

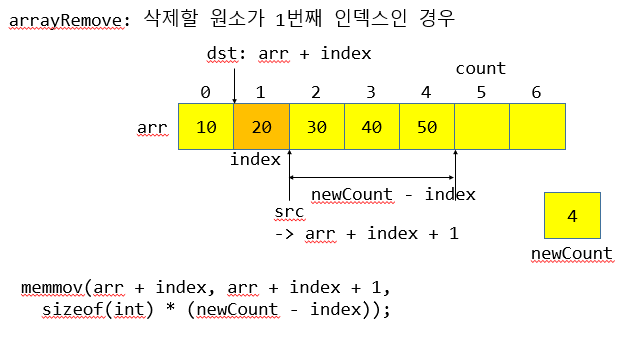
**<<<<<삭제>>>>>**

자료구조에서의 삭제는 물리적으로 지우는것이 아닌, 자료구조 내에 그 데이터를 더 이상 유지하지 않는다는 개념이다. 삭제된 데이터는 사용자에게 전달하여 사용자가 처리하도록 해야한다.

🡺0527 c.c



만약 마지막 변수를 제거하는 상황이라면 삭제할 필요없이 count값만 하나 내려주면 된다. 어차피 나중에 변수 추가가 일어나면 어차피 덮어씌워질 것이다.



**<<<<<동적배열>>>>>**

배열을 동적으로 늘리는 개념은 기존의 배열을 더 큰 배열에 복사를 한뒤 기존 배열을 없애는 느낌이다.

🡺d.c

arrayCreate를 바꾼 코드(#define을 통한)

배열의 크기가 자동으로 증가될 경우, 배열의 크기를 사용자로부터 입력받을 필요가 없다.

🡺e.c

배열이 꽉찻을경우에 더 추가된 데이터가 있을경우 동적으로 배열의 길이를 추가(x2)하는 코드

arrayAdd()수정. array구조체 안의 contents를 두배로 늘려준후 복사하고 원본을 복사하면 된다.

그러나 비효율적이다..(주석처리함) 그래서 realloc을 쓰면 좋다!

만약 realloc이 실패하면 null을 반환하는데 그러면 array->contents 에 null이 들어가므로 메모리 누수가 발생한다. 그래서 임시포인터를 사용하여 따로 빼서 만들어준후 if조건문을 통과했을 때 대입해주자.

그리고 arrayInsert에도 원래는 count와 size가 같으면 오류를 출력해주었었는데 이제는 동적배열로서 가능하므로 위의 코드를 추가해주면 꽉찼을때도 insert가 가능해진다.

🡺f.c

그러나 위 코드는 자원 할당 정책이 중복되어있다는 단점이 있다. 이는 유지보수를 어렵게 하고 (그래서 할당하는 것들은 한곳에서 몰아서 하는게 좋다) 버그가 발생할 확률이 높아진다.

이를 해결하기 위해 자원 할당 정책을 한곳으로 모으자

->increaseSize ->사용자가 원하는 만큼의 size만 증가시켜주자!

만약 size – 50 count – 6이면 굳이 동적할당 할 필요가 없다. 그냥 넣으면 되니까! 그래서 이러한 경우에는 increaseSize를 실패햇다 가정하고 return -1 해주자.

내일 오전에 다시!!