

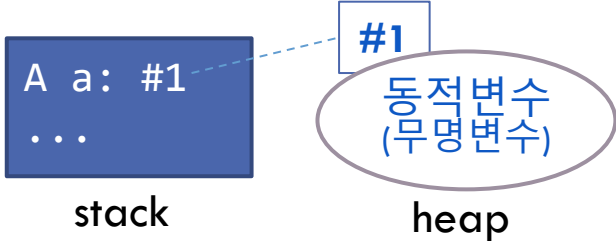
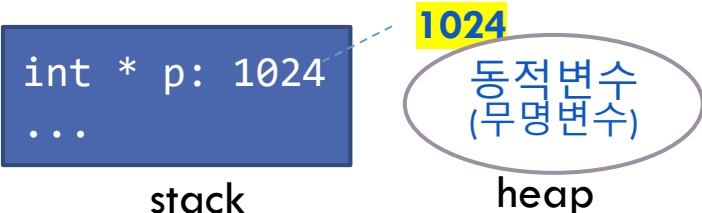
Java를 알고 C배우기

컴퓨터프로그래밍3

week 1 1-2 동적 메모리-**malloc**

2022.1학기
충남대 조은선

C의 동적 변수 만드는 방법

		Java	C
생성	동적 변수 생성 명령	<code>new()</code>	<code>malloc</code>
	생성 시 전달되는 것	클래스 이름 (, 생성자 인자)	생성될 메모리 크기
	예	<code>new A()</code> // A는 클래스 이름	<code>malloc(sizeof(int))</code> <code>malloc(24) ...</code> // 직접 바이트 수를 적기도 함
	동적 변수 초기화	생성자	x (안함)
	생성 후 리턴 값	클래스 A타입 객체의 OID	<code>void *</code>
	생성 결과 활용	<code>A a = new A();</code>	<code>int * p = (int *) malloc(sizeof(int));</code>
삭제	삭제 명령	x (불가능)	<code>free()</code>
	삭제 주체	Garbage collector	개발자 (프로그램에 명시)
Stack 변수와 관계			

동적 변수 만들기

- ▶ malloc은 stdlib.h를 include해야 사용 가능

`#include <stdlib.h>`

- ▶ malloc의 타입 : `(void *) malloc(size_t size);`

참고: size_t는 결국 unsigned int 임 (#define size_t unsigned int)

- ▶ malloc 활용 예

```
int* p;  
*p = 3;      // 이미 배웠음. 변수 p의 값을 주소로 해서 3을 넣고 있으나.... Error(가 만나와도 에러!)
```

```
int num;  
int *p = &num;  
*p = 3;      // 이미 배웠음. 포인터의 활용. O.K.
```

```
int* p;  
p = (int*)malloc(sizeof(int));    // (int*)는 형 변환 연산자  
*p = 3;                          // O.K.
```

malloc 함수의 형 변환

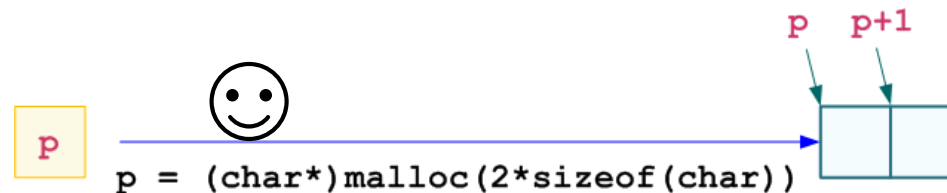
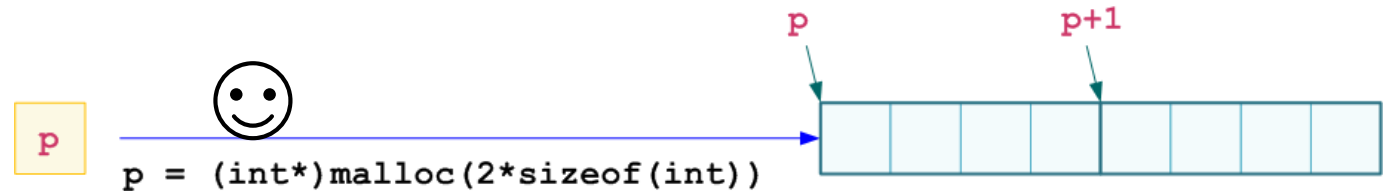
▶ void * 타입 변수

- ▶ 어떤 타입 내용의 주소라도, 주소라면 모두 담을 수 있는 변수
- ▶ 하지만, 해당 포인터 변수의 값을 주소로 하여 내용을 읽어야하는데
- ▶ 얼마큼 읽을지가 명시되지 않는다
- ▶ 즉, 형변환 없이 바로 내용을 가져오면 오류!

```
int num = 10;
void *p = &num;
p++; // Error!
printf("%d", *p); // Error!
printf("%d", *(int*)p); // O.K.
```

▶ malloc의 형변환

- ▶ 리턴 타입이 void* 타입이므로
- ▶ 제대로 사용하려면 형 변환이 필수임
- ▶ 가리키는 대상을 몇 바이트 단위로 읽을지?



포인터 생성-사용 예

생성 패턴

```
int * ptr1 = (int *) malloc(sizeof(int));
double * ptr2 = (double*) malloc(sizeof(double));
int * ptr3 = (int*)malloc(sizeof(int)*7);
double * ptr4 = (double *)malloc(sizeof(double)*9);
```

접근 패턴 1

```
int *ptr1 = (int*) malloc(sizeof(int));
*ptr = 20;
printf("%d\n", *ptr1);
free(ptr1);
```

접근 패턴 2

```
int *ptr2 =(int*)malloc(sizeof(int)*7);
int i;
for(i= 0; i < 7; i++)
    ptr2[i] = i+1;
for(i=0; i <7; i++)
    printf("%d ", ptr2[i]);           // 결과 1 2 3 4 5 6 7
free(ptr2);
```

free: 동적변수 삭제

- ▶ malloc 된 변수의 free

- ▶ 할당된 메모리 공간은 프로그램 끝날 때 까지 프로그램에 귀속
- ▶ 더이상 쓸모가 없어진 경우에 free
- ▶ 예) `int * p = (int *) malloc(sizeof(int)*2000);`

...

`free(p);` // 언제부터 쓸모 없어지는지는 개발자가 가장 잘 안다

- ▶ free를 안 해도 되는 것 아닌가?

- ▶ 메모리 리소스 관리 문제
 - ▶ malloc에 인자로 전달하는 메모리 크기가 매우 클 수도 있고 (많은 경우)
 - ▶ malloc 호출 수가 많을 수도 있음
- ▶ 최소한 프로그램이 수행되는 동안에는 해당 메모리를 다른 곳에 활용할 수 없음
cf. memory leak, out of memory error

Quiz

```
char * read_username( ) {  
    printf("What's user name?");  
    scanf("%s", name);  
    return name;  
}  
int main() {  
    char *name1, *name2  
    // char* 타입 포인터 변수 2개 선언/정의  
  
    name1 = read_username();  
    printf("name1: %s \n", name1);  
    name2 = read_username();  
    printf("name2: %s \n", name2);  
    free(name1);  
    free(name2);  
}
```

왼쪽 read_username()은 이름을 입력 받아 리턴하는 함수이다. 밑줄 그은 곳에 입력했을 때 정상적으로 작동하는 경우는? 이유는?

- (1) char *name =
 (char*)malloc(sizeof(char)*30);
- (2) char name[80];
- (3) (1), (2) 둘다 정상적으로 작동
- (4) (1), (2) 둘다 오류