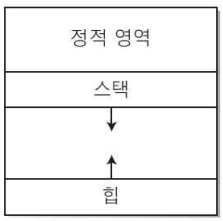
할당

- 기억장소 할당
  - 변수에 메모리 공간을 바인딩하는 과정
- 회수
  - 변수로부터 바인딩이 해제된 메모리 공간을 가용 공간으로 돌려주는 과정
- 수명
  - 변수가 특정 메모리 주소에 바인딩되어 있는 시간
  - 변수의 수명은 변수가 메모리 공간에 바인딩될 때 시작되며 회수될 때 종료
- 변수의 할당은 수명에 따라 정적 할당, 스택 기반 할당, 동적 할당으로 분류

할당

- 변수 할당과 관련된 메모리 구조기억장소 할당
  - 정적 영역은 크기가 정해져 있음
  - 스택과 힙은 각기 반대 방향으로 영역이 성장할 수 있다.

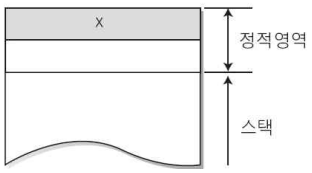


할당

- 정적 할당
  - 변수에 메모리 공간이 정적으로 할당되는 것
  - 한번 할당되면 프로그램 실행이 종료될 때까지 할당 상태가 그대로 유지
  - 정적 할당이 이루어지는 메모리 공간은 정적 영역

예) 변수 x가 전역 변수로 정적 할당되는 예

```
int x;
int main(void)
{
        ⋮
```



할당

- 정적 할당을 하는 지역 변수의 예

```
01  #include<stdio.h>
02
03   int func(void)
04  {
05        static int count = 0;
06        count++;
07        return count;
08  }
09  int main(void)
10  {
11        int i;
12        for (i=0; i<10; i++)
13             printf("%d ", func());
14        return 0;
15  }
```

count 변수 : 지역변수, 정적할당
➔ func 함수가 종료되어도 count는 메모리에서 회수되지 않음

할당

- 스택 기반 할당
  - 자동 할당(automatic allocation)이라고도 함
  - 변수의 타입은 정적으로 할당되지만 메모리 공간은 실행 시간 중에 할당
  - 스택 기반 할당이 이루어지는 메모리 공간은 스택

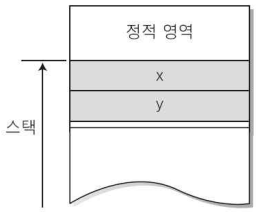
할당

- 스택 기반 할당을 하는 지역 변수의 예

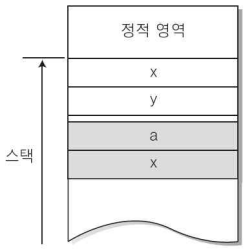
```
01  void func(void)
02  {                          스택 기반 할당을 하는 지역변수
03      int a;
04      double x;
        ...
05  }
06  int main(void)
07  {
08      int x, y;
09      func();
        ...
10  }
```

할당

- 메모리에 할당되는 과정
  - main 함수의 x와 y 할당

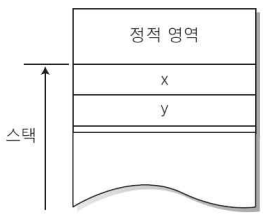


  - func 함수의 a와 x 할당

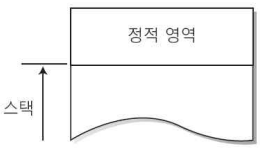


할당

- 메모리에 할당되는 과정
  - func 함수의 a와 x 회수



  - main 함수의 x와 y 회수



할당

- 동적 할당
  - 명시적인 명령어에 의해 실행 시간에 할당
  - 동적 할당이 이루어지는 메모리 공간은 힙(heap)
  - 동적으로 할당된 영역은 포인터나 참조 변수를 통해서 참조 가능

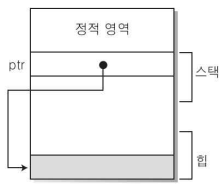
  - 예) C
    - malloc 함수와 free 함수 제공
    - stdlib.h 파일을 include 시켜야 사용 가능
    - 동적으로 메모리를 할당하고 회수하는 C 코드

```
01   int *ptr;
02  ptr = (int *)malloc(sizeof(int));
03  *ptr = 20;
04   free(ptr);
```

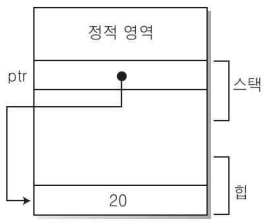
할당

- 동적 할당 과정

  - ptr이 동적으로 할당된 영역을 가리킴

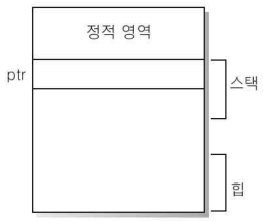


  - ptr이 가리키는 영역에 20을 배정



할당

- 동적 할당 과정
  - 동적으로 할당된 영역의 회수



  - C++
    - 동적 할당과 관련된 연산자 제공 : delete, new

```
01  int *ptr;
02  ptr = new int;
03  *ptr = 20;
04  delete ptr;
```

이름 상수

- 이름 상수
  - 프로그램 전반에 걸쳐 고정된 값을 가지는 식별자
  - 변수와는 달리 값이 변경될 수 없음
  - 판독성과 프로그램의 신뢰성을 증진
  - 판독성 향상 예

```
area = 5 * 5 * 3.14159;
```

➔ 상수 pi 사용하여 판독성 향상

```
const double pi = 3.14159;
area = 5 * 5 * 3.14159;
```

이름 상수

- 신뢰성 향상 예

```
int score[100];
    :
for (i=0; i<100; i++) {
    :
 } average = sum / 100;
```

```
const int number = 100;
int score[number];
    :
for (i=0; i<number; i++) {
    :
} average = sum / number;
```

➔ 학생 수를 의미하는 이름 상수 number 사용

- 학생 수가 변경되어도 이름 상수를 선언하는 문장만 수정하면 된다.
  - const int number = 200;

이름 상수

- Ada : constant로 이름 상수 선언

```
number: constant integer := 100;
```

- Java : final로 이름 상수 선언

```
final int number = 100;
```

- Ada, C++, Java는 이름 상수에 값을 동적으로 바인딩하는 것을 허용

```
const int current = (int) time(0);
```